TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JANNE HARJU

PLANT INFORMATION MODELS FOR OPC UA: CASE COPPER REFINERY

Master of Science Thesis

Examiner: Hannu Koivisto
Examiner and topic approved in the Council meeting of Depart of Automation Science and Engineering on 5.11.2014

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
HARJU, JANNE: PLANT INFORMATION MODELS FOR OPC UA: CASE COPPER REFINERY
Diplomityö, 59 sivua
Tammikuu 2015
Pääaine: Automaatio- ja informaatioverkot
Tarkastaja: Professori Hannu Koivisto
Avainsanat: OPC, UA, ISA-95, CAEX, mallinnus, ADI

Isojen laitosten informaation rakenne on usein hankala ihmisen hahmottaa silloin, kun kaikki tieto on yhdessä listassa. Jotta laitoksen informaatiosta saadaan parempi kokonaiskuva, tarvitsee se mallintaa jollakin tekniikalla. Kun laitoksen mallintaa hierarkkisesti, pystyy sen rakenteen hahmottamaan helpommin. Lisäksi, kun malliin saadaan tuotua mittaussignaaleja ja muuta mittauksiin ja muihin arvoihin liittyvää dataa, saadaan se tehokkaaseen käyttöön.

Nykyään paljon käytetty klassinen OPC tuo datan käytettäväksi ylemmille laitoksen tasoille. OPC tuo mittaukset pelkkänä listana, eikä se tuo mittausten välille mitään metatietoa. Tässä työssä on tarkoitus tutkia, miten mallintaa esimerkki laitokset käyttäen klassisen OPC:n seuraajaa OPC UA:ta. Työn tavoitteena on tutkia eri OPC UA:n mallinnustyökaluja, menetelmiä ja itse mallinnusta. Työssä tutustutaan OPC UA:n lisäksi ISA-95:n ja sen OPC UA- malliin ja miten tätä mallia pystyy käyttämään oikean laitoksen mallinnuksessa hyödyksi. ISA-95 lisäksi tutustutaan toiseen malliin nimeltä CAEX. CAEX on alun perin suunniteltu suunnitteludatan tallentamiseen standardoidulla tavalla. CAEX:sta on olemassa myös tutkimuksia OPC UA:n kanssa käytettäväksi ja tässä työssä tutustutaan, onko mallinnettavien tehtaiden kanssa mahdollista käyttää CAEX:ia hyödyksi. Virallista OPC UA CAEX- mallia ei ole vielä tosin julkaistu. Lisäksi tutkitaan, miten CAEX- malli ja ISA-95- malli eroavat toisistaan. Työssä mallinnetaan osittain yksi kuparijalostamo ja yksi elektrolyysilaitos. Työssä lisäksi pohditaan, miten kyseiset laitokset saadaan mallinnettua ja miten malleja pystytään järkevästi ylläpitämään muutostilanteissa.

ISA-95- mallin käyttö onnistui hyvin ja esimerkki tehtaat saatiin mallinnettua tarvittavalla tarkkuudella OPC UA avaruuteen. Työkalussa jota käytettiin tehtaiden mallinnuksessa, tuli vastaan joitain ongelmia, jotka työkalun kehittäjät korjasivat. Lisäksi saatiin suunniteltua, miten klassisen OPC:n kautta tai muista UA servereistä saadaan tuotua dataa.

Työssä tutkitaan myös erilaisia arkkitehtuuriratkaisuja. Työssä lopulta päädyttiin kahteen vaihtoehtoon. Toinen on helppo konfiguroida ja ottaa käyttöön klassista OPC:tä käyttäen. Toinen, joka on vaikeampi konfiguroida ja ottaa käyttöön, mutta jolla on paremmat tulevaisuuden mahdollisuudet OPC UA:ta täysin hyödyntäen.

Tuotannonohjaus tyyliseen käyttöön ISA-95- malli on CAEX- mallia parempi. CAEX- mallia kannattaa käyttää silloin, kun suunnittelutyökalut tukevat CAEX- mallista dataa ja lopputulos tulee SCADA- tason käyttöön. Lisäksi, jos tarvitaan ISA-95:n sisältämiä rakenteita ja tietotyyppejä, ei CAEX:a voida käyttää.

Työssä testattiin lisäksi, miten OPC UA Analyzer Device Integration- mallia voitaisiin käyttää raekoko analysoijan kanssa. Selvisi, että malli soveltuu erittäin hyvin kyseiseen tehtävään. ADI- mallin rakennuspalikat mahdollistavat analysoijan mallintamisen tarpeeksi joustavalla tavalla.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Engineering
HARJU, JANNE: PLANT INFORMATION MODELS FOR OPC UA: CASE COPPER REFINERY
Master of Science Thesis, 59 pages
January 2015
Major: Automation- and information- networks
Examiner: Professor Hannu Koivisto
Keywords: OPC, UA, ISA-95, CAEX, modelling, ADI

It is not unusual that there is not always a good overall picture of all the information a big plant contains, especially when we go to single equipment. To get a better overall picture from the plant it should be modelled by some technique. When a plant is modelled hierarchically it is easier to get a better understanding of the plant. After you get measurement signals and other data to the model the model can be used effectively.

Currently popular technique, classic OPC, can bring data from the equipment. The data in OPC is just a plain list without any metadata. The goal in this thesis is to research how to model the example plant using OPC UA which is a successor of the classic OPC. The main three goals in this thesis are to research different OPC UA modelling tools, methods and modelling. Thesis also consists of getting to know ISA-95 and its OPC UA model and how it can be used in modelling a real life plant. After getting to know ISA-95, there is a section about the CAEX- model. CAEX was initially designed to store engineering data from the engineering tools. There are some researches how it can be used with OPC UA. The official OPC UA CAEX- model is not yet released but some research has been done in Germany. In this thesis those researches will be used to check if the CAEX- model is a good choice in this situation. Next ISA-95 and CAEX models will be compared. For testing these models few different plants will be modeled. In this thesis it is researched how to model a plant and how to get the needed engineering data to succeed in this task.

There were some problems when modelling plant models with the chosen modelling tool but those problems were solved by the tool distributor. After the fixes the plant was model successfully over the ISA-95-model. It is also covered how to get real-time data from a real plant into the OPC UA model using the classic OPC or other UA servers.

One part of the thesis involves few alternatives for the architecture design. After researching there are few possible architecture choices, one of which is easy and cheap to configure with classic OPC and another which is harder to configure but which have better future possibilities.

The CAEX- model was researched and the result was that it wasn't the best choice to model a plant in this thesis. It will be better to use the ISA-95- model to model a MES-level plant instead of CAEX. CAEX can be used for example with SCADA systems if engineering tools can offer CAEX- data.

OPC UA Analyzer Device Integration model also was tested for particle size analyzer. Result was that model can be used well with this analyzer, although working server was not developed in this thesis. Building blocks in ADI model give good foundation to make analyzer device model.

PREFACE

This thesis was written between spring 2014 and winter 2014. I got a lot of support and guidance from Martti Larinkari and Ahti Rossi at Outotec and also from Aki Kössilä at Delta-Enterprise. My special thanks go to my wife Päivi, your support means a lot to me. I also thank my family for all the support during my life, with your help I was able to get this far.

Nurmijärvi, December 17th. 2014
Janne Harju

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

.NET	Software framework developed by Microsoft
ADI	Analyzer Device Integration
AM	Asset Management
APM	Anode Preparation Module
ASW	Anode Scrap Washer
B2MML	Business To Manufacturing Markup Language
C#	Programming language mainly used in .NET framework
CAEX	Computer Aided Engineering Exchange
CDT	Core Data Type
CMMS	Computerized Maintenance Management Systems
COM	Component Object model
CSM	Cathode Stripping Machine
DA	Data Access
DCOM	Distributed Component Object model
DCS	Distributed Control System
DI	Device Integration
DLL	Dynamic-link library
EDDL	Electronic Device Description Language
ER	Electrorefining
ERP	Enterprise resource planning
EW	Electrowinning
FDI	Field Device Integration
HTTP	Hypertext transfer protocol
ISA-95	International Society of Automation standard 95
Java	Programming language
KPI	Key Performance Indicator
LIMS	Laboratory Information Management Systems
MES	Manufacturing execution system
MOM	Manufacturing Operations Management
OLE	Object Linking and Embedding
OOE	Object-Oriented Encapsulation
OPC	OLE for process control, or now days Open Platform Communications
OPC UA	OPC Unified Architecture
OSI	Open Systems Interconnection
PLC	Programmable Logic Controller
PPR	Products, Process and Resources
PSI	Particle Size Analyzer
SCADA	Supervisory Control And Data Acquisition
SDK	Software Development Kit
SOA	Service Oriented Architecture

SQL	Structured Query Language
TCP	Transmission Control Protocol
TFM	Tank Farm Management systems
TIMS	Tankhouse Information Management System
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WMS	Warehouse Management Systems
WSDL	Web Service Description Language
XML	Extensible Markup Language

TABLE OF FIGURES

1 INTRODUCTION

The goal of this thesis is to test different OPC UA information models with different tools and make a general guide how the modelling can be done. To reach this goal two example plants are modelled in the decided precision using selected techniques and selected tools. This will help to understand the hierarchy of the big production plants. There is no good overview of the plants so this will be needed. The situation today is that most of the data which one can get from the plant is read via classic OPC which makes all data flat. There are no hierarchies between different signals and there is no way to add more information and metadata into a signal.

In the last 20 years classic OPC has been in de facto role when connecting PC software to the industrial devices like Programmable Logic Controllers (PLC). Classic OPC's successor, OPC Unified Architecture (OPC UA) is becoming more and more popular in new devices because of its good scalability and good modelling capabilities. Modelling in this thesis is done by using OPC UA.

The ISA-95 companion specification is one of the many companion specifications which are done for the OPC UA. The ISA-95 itself is a lot more than just an OPC UA model. The ISA-95 is a 5 part specification for integrating Manufacturing Execution System (MES) and Enterprise Resource Planning (ERP) systems. It includes several different resource models, which are Equipment, Physical Asset, Material and Person. The focus in this thesis is on the Equipment- and Physical asset- models.

Computer Aided Engineering Exchange (CAEX) will be introduced and also compared with the ISA-95 model. CAEX model specification for the OPC UA is under development at the moment but use of CAEX with OPC UA will be researched.

Modelling can be tested and models can be done with different modelling tools. Modelling tools are made for developing OPC UA address space with graphical user interfaces. In one part of this thesis few selected tools will be tested and most suitable will be used in the implementation part of this thesis.

1.1 Objectives

The first objective is to get familiar with the used techniques. The techniques which are necessary to get familiar with are OPC, OPC UA, ISA-95 and CAEX, but the main focus will be on the OPC UA and the ISA-95 OPC UA model. Other objectives are how to model plants and devices in the plant and what kind of data can be set into the equipment model. To model these devices and plants it is necessary to know what kind

of tools there are. One goal is to get familiar with these selected tools. Result of this thesis will be a general guide for modelling with OPC UA and especially with the ISA-95 OPC UA model.

The main objective in this thesis is to model two example plants using the ISA-95 OPC UA model. To model these plants it is needed to make some example equipment that are used when modelling the hierarchy models of the plants. When modelling the equipment it is necessary to know how to do such a thing and what information is needed to manage in this task. In one part of this thesis the example plants are introduced. That helps to understand what will be modelled.

The last goal is to design how this OPC UA model can be used in factories in real life and how data can be connected into models from the older information systems. These new UA models also introduce more ways how to use the information and these possibilities are also researched.

The resources that are used in this thesis are several different specifications, few research papers and few master of thesis works. Some information was also found from the web sites. The PI diagrams of the process were used to understand the structure of the process.

1.2 Outline of the thesis

Thesis starts with the background check where all used techniques are introduced. The next chapter consists of how to model devices and plants and what kind of design tools there are. After that there is a chapter where the example plants are introduced. The fifth chapter is about the implementation including examples how to implement the plant's information model with the selected tool. The fifth chapter also consists of how the OPC UA server can be integrated with the old systems. It will also be researched how the analyzer device integration model can be used to model a particle size analyzer. The sixth chapter is about results and discussions, it is discussed how well goals were achieved. The last chapter is about conclusions where the work is summarized and the possible next development steps are discussed.

2 BACKGROUND

Modelling will be done in this thesis using OPC UA and several different model specifications. Term modelling in this thesis means making general types about the typical equipment used in plants. These types also include typical measurements and variables that equipment usually contains. Next phase of the modelling is to model plants. This means making a plant hierarchy using these just modelled equipment types. Target of the thesis is to model 2 different plants and one analyzer. First reason for modelling with OPC UA is that it helps to gather the data from different plants into centralized server. When each plant uses same equipment types it is much easier to compare efficiency and other values of the equipment. This can help to make new business opportunities which can be offered for the customers. Other reason for the modelling is to make general practices of how things should be done. Type models also help to understand structure of the complex equipment whereas the hierarchical model helps to understand the hierarchy of the plants. These two together assist to understand relations between signals and measurements. This understanding is needed because different measurements and variables are used in different levels.

Selected techniques of this thesis are OPC UA, ISA-95, ADI and CAEX. OPC UA is selected because it is very flexible for any type of modelling. Its ancestor classic OPC is de facto technique in today's plants. OPC UA is backwards compatible with classic OPC servers. It is not certain that OPC UA will become a major technique in the field it solves, but it is the most hyped integration technique out there. The ISA-95 was selected because it is generally used when integrating manufacturing execution systems. There is also an ISA-95 companion specification for the OPC UA which can be used in the OPC UA server. This companion specification does not cover all ISA-95 features yet. It only covers resource models. That is why this thesis is focusing only on resource part of the ISA-95 specification. Other companion specification that is researched in this thesis is analyzer device integration (ADI). This was selected because one of the example plants include several particle size analyzers with which the ADI model can be tested and used with. CAEX was selected to research how it could be used in modelling plants and how it can be integrated with OPC UA. CAEX is based on an XML file format. It is used in exporting and importing data from the design software for further use.

OPC UA has been researched a lot. There are several thesis where OPC UA is in a lead role like Laukkanen's and Palonen's thesis [Laukkanen 13] [Palonen 10]. General use case with OPC UA is still to make a gateway between two classic OPC servers. Some

plants have started to use OPC UA, but it is not yet in general use. OPC UA is quite complex and that might be the reason why it is not yet in wide use. Complexity might be result of trying to resolve every problem in the field. The ADI model has been researched by few companies and it is proven to be useful model with analyzers [Luth 10]. The CAEX integration into OPC UA has been researched in Germany and the CAEX companion specification is under development there [Schleiben 10]. Jouko Virta has also already researched how OPC UA and ISA-95 can be used when integrating ERP and MES level systems [Virta 10, p.1].

This chapter introduces to the used techniques: ISA-95, OPC, OPC UA, ADI, CAEX and UML. These techniques are researched and introduced in a depth that is necessary to understand the implementation part of the thesis. We will start introducing ISA-95 in general. After this OPC and OPC UA standards are explained. Also several OPC UA companion specifications are introduced. Next CAEX will be outlined and compared with the ISA-95 standard. At the end of this chapter the Unified Modelling Language (UML) will be presented and its notation will be compared to OPC UA's notation. UML is used to present several different models in this thesis.

2.1 The ISA-95 Enterprise Control Systems Standard

The ISA-95 standard is developed by the International Society of Automation. It is made to standardize manufacturing operations and control functions. It standardizes interfaces between the Enterprise Resource Planning (ERP) and the manufacturing operations level. This reduces integration effort between different products which implements ISA-95. [ISA-95.01 10, p.13]

The ISA-95 specification consists of five parts. These are Models and Terminology, Object Model Attributes, Activity Models of Manufacturing, Objects and attributes for manufacturing and Business-to-Manufacturing. Part six is still under development while writing this. Focus in this thesis will be on part 2 which is Object Model Attributes. But we will take an overlook on the whole specification to be able to understand a part of it. [ISA-95.01 10, p.8]

Development of the specification started at 1995 at the Purdue University. The ISA-95 specification has been developed over 15 years. It is designed for industrial use. The main focuses on this specification are the interfaces between level 3 and level 4 in the hierarchy model [ISA-95.01 10, p.21]. These levels and the hierarchy model will be defined later in this chapter. Levels are Enterprise resource planning system and Manufacturing Operations Management (MOM) system. The ISA-95 specification uses UML diagrams for describing different models. [Ninety-five]

MOM level systems are defined in ISA-95 specification. They are presented in Table 2-1. They help to coordinate and control the plant. They are used for example to data collection and acquisition, quality operations management, process management and performance analysis. [OPC UA ISA-95 13, p.7]

Table 2-1: Manufacturing Operation Management systems[OPC UA ISA-95 13, p.7]

Abbreviation	**Full name**
MES	Manufacturing Execution System
LIMS	Laboratory Information Management Systems
WMS	Warehouse Management Systems
TFM	Tank Farm Management systems
AM	Asset Management
CMMS	Computerized Maintenance Management Systems

There is already de facto standard interface implementation of the ISA-95 standard called Business To Manufacturing Markup Language (B2MML). It is an open source XML implementation of the ISA-95 and IEC 62264 standards. The ISA-95 OPC UA model specification uses this interface. [OPC UA ISA-95 13, p.1]

The ISA-95 standard introduces plants with five levels, which are described in Figure 2-1. This thesis is focusing on level 3.

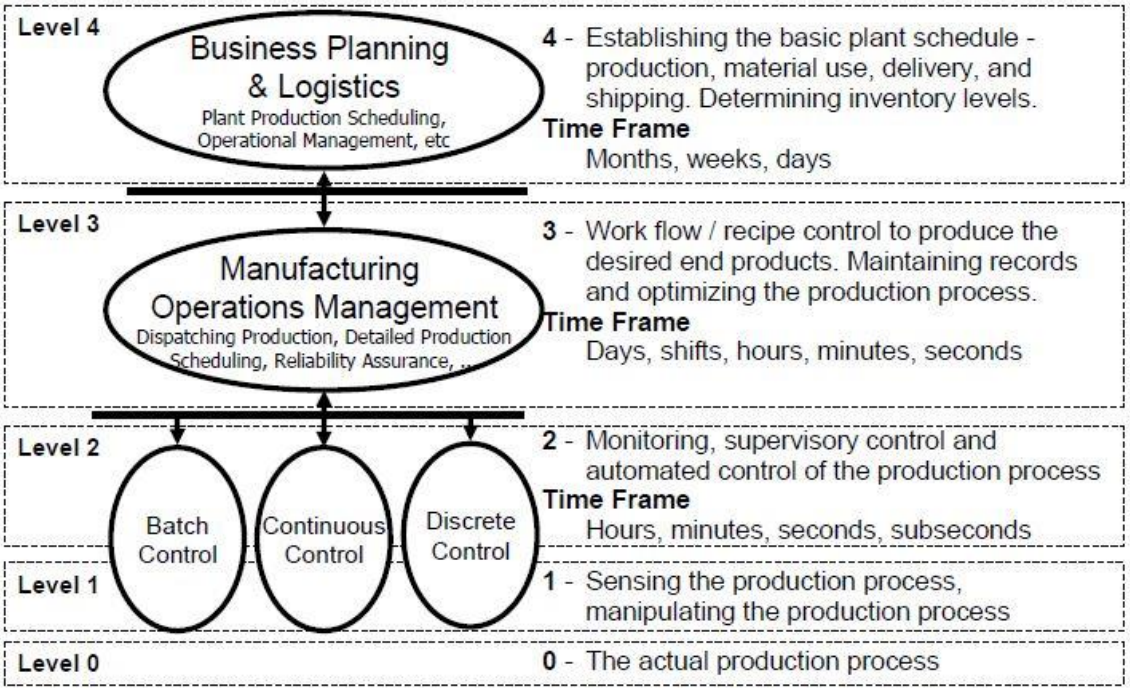


Figure 2-1: Functional hierarchy [ISA-95.01 10, p.21]

The ISA-95 standard describes several different models and activities. Resource models are one of these models and they are used in this thesis. Resource models are Personnel, Material, Equipment and Physical Asset. These models contain data which is used by level 3 and level 4 activities.

The ISA-95 specification defines the relation between Equipment and Physical Asset, which is *implemented by*. This relation describes which physical asset implements certain equipment. There are no other relations between resource models. These models declare how to describe resources. In each model there are corresponding resource definition, resource property, resource classification definition, resource classification property and resources capability test result. Figure 2-2 present Equipment model and Physical asset model relationship. [ISA-95.02 10, p.21]

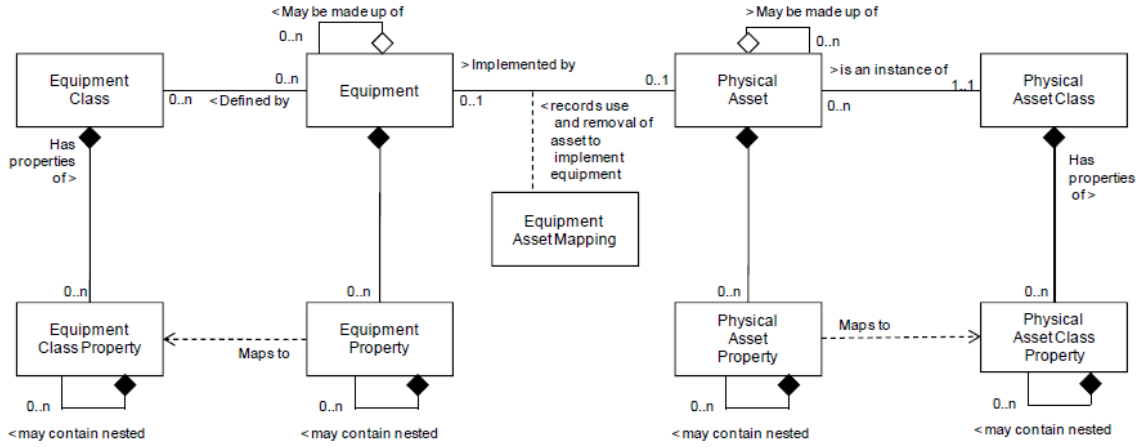


Figure 2-2: ISA-95 Equipment and physical asset relationship [ISA-95.02 10, p 41]

This thesis is focusing on equipment and physical asset models and their relations. But also person and material models are used in some examples. Equipment and physical asset can hold information of the asset assignments. This helps to record which Physical asset implements which Equipment. There is also information when this assignment has been used. This can be extended if needed [OPC UA ISA-95, p.56]. Plant can be modeled and different levels can be set for equipment if equipment model is in use. Equipment levels in the ISA-95 standard are divided to four categories depending on the industrial field. Industrial fields are batch, continuous, discrete and storage. Equipment levels at the top are the same for all industrial fields: enterprise, site and area. Lower levels are different depending on the industrial field. This is presented in Figure 2-3. [ISA-95.01 10, pp 25-30]

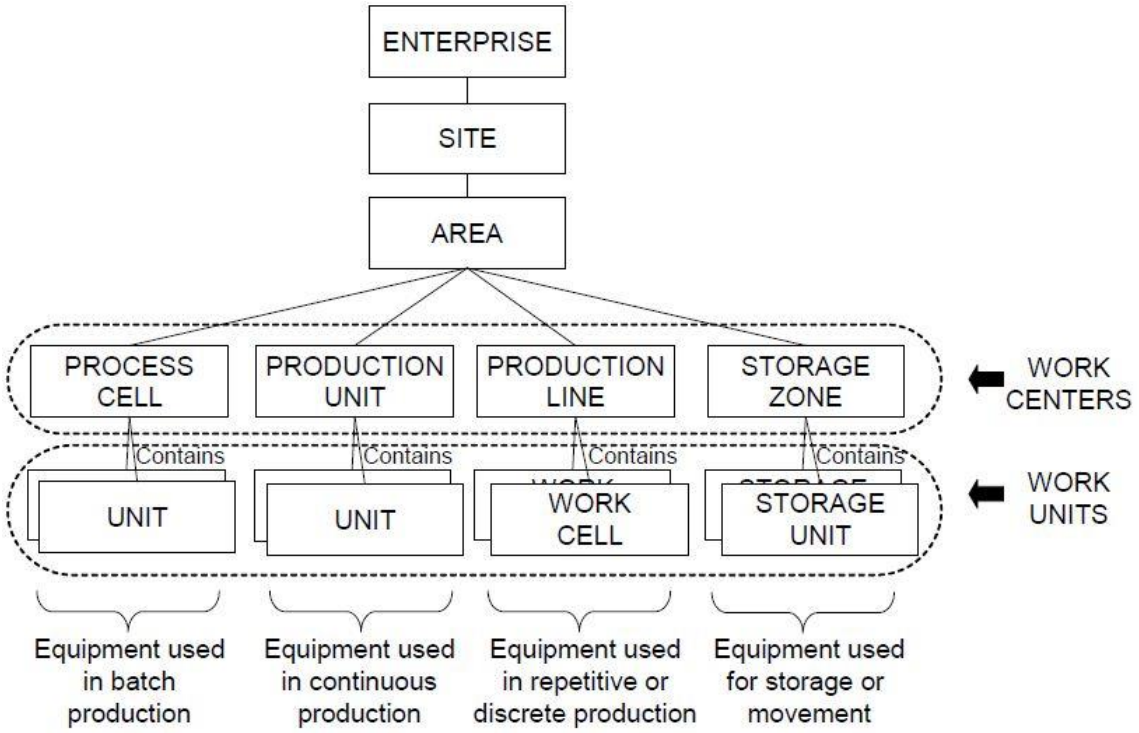


Figure 2-3: ISA-95 role based equipment hierarchy levels [ISA-95.01 10, p 28]

This thesis is focusing on continuous production and batch production fields, because example plants are continuous process and batch process. Levels which are used later are: Enterprise, Site, Area, Production Unit, Process cell and Unit.

The ISA-95 standard defines activities which are presented in Figure 2-4. Specific activities are planned to work with specific equipment levels. Level 4 activities are planned to work with enterprise and site equipment. Level 3 activities are planned to work with site equipment but also with area, production unit and unit equipment. Activities are not of great interest in this thesis but it is good to know what these Equipment and Physical assets are dealing with. [ISA-95.01 10, p 26]

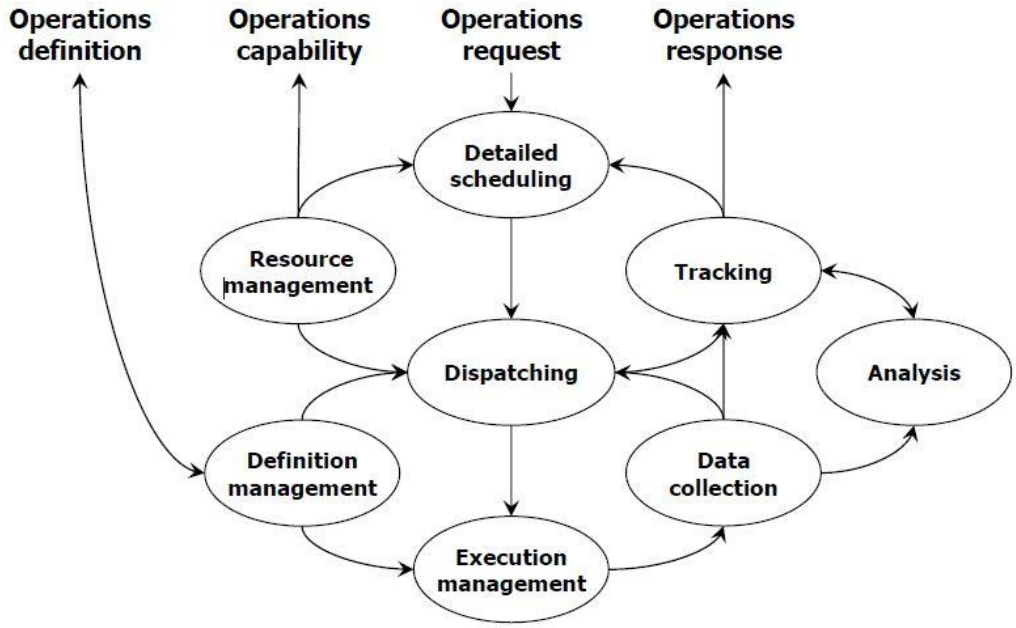


Figure 2-4: ISA-95 Generic Manufacturing Operations Management Activities [ISA-95.03 13, p 18]

Activities at level 4 are for example collecting and maintaining overall energy usage and planning production capacity. Level 3 activities are for example collecting and maintaining area data about product quality, spare parts and raw materials. Level 2 activities are more like local control of processes. Level 1 activities are to monitor and manipulate processes using manual sensing, sensors and actuators. Majority of the activities which are specified in the ISA-95 specification are working in level 3. Descriptions of all activities are quite abstract, because use cases of the ISA-95 standard varies a lot between different plants. [ISA-95.01 10, pp. 22-23]

Figure 2-4 presents a set of the activities. Certain activity can use collection of activities. For example *Production dispatch*- activity uses detailed scheduling, tracking, dispatching and resource management. Some activities which are presented in Figure 2-4 include several inner activities. Below these activities are level 1 and 2 activities. [ISA-95.03 13, pp. 16-18]

2.2 Open Productivity & Connectivity (OPC)

Development of the OPC standard started because there was a need for common interface for devices like Programmable Logic Controllers (PLC). The classic OPC has only a small part in this thesis but it is good to understand the history of the OPC UA. That is why OPC is introduced shortly in this chapter.

Abbreviation OPC comes at first from OLE for Process Control. Here OLE stands for Object Linking and Embedding. Nowadays meaning of the OPC has been changed and it is now called Open Productivity and Connectivity. Some also use the term Open Platform Communications. Behind the OPC standard there is an OPC Foundation which consists of over 450 companies. Main force behind the OPC has been Microsoft. The

First OPC standard was released 1996. OPC is based on Microsoft Component Object Model (COM) and Distributed COM (DCOM) technologies. For that reason OPC is tied to Microsoft's Windows operating systems. Platform dependency is one of the biggest flaws of OPC today. OPC Data Access (DA) is usually called classic OPC. In addition to classic OPC there are several other OPC specifications. These specifications are listed in Table 2-2 below. [Mahnke 09, p.1]

Table 2-2: Classic OPC specifications

Abbreviation	**Full name**
DA	Data Access
A&E	Alarms & Events
HDA	Historical Data Access
XML-DA	XML Data Access

In addition to these there are also couple extensions to Data access (DA) specifications: Complex Data, Batch and Data eXchange.

Use of the OPC is described in Figure 2-5. Device manufacturer creates its own OPC driver for their devices. OPC server can connect to a device like PLC using this driver. OPC client can connect to OPC server after OPC server is configured. Using that client, programmer can connect to any OPC server. Configuring OPC means that, driver will be configured so that the OPC server can connect to the device. After that tags will be added to that driver. Tag consists of at least address, data type, tag name and update rate.

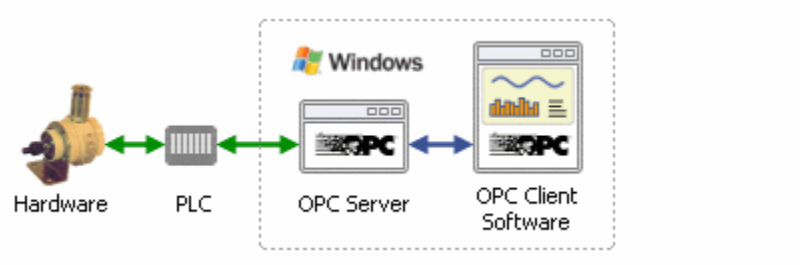


Figure 2-5: Usual use of OPC [OPC DataHub]

Because of the used techniques and the lack of security and scalability of the OPC, OPC Foundation started to develop next generation integration technique called OPC Unified Architecture (OPC UA). Next subchapter introduced this technique.

2.3 OPC Unified Architecture (UA)

OPC Unified architecture (UA) was made to expand classic OPC features and group classic OPC specifications into one specification. First version of the OPC UA was released in 2006 after 3 years of development. It was developed for several reasons. Classic OPC is dependent on Microsoft operating systems, because it uses Microsoft COM and DCOM technologies. This dependency is not desirable anymore. Field devices have become more intelligent than ever and they can host little servers by themselves. Plat-

form independence can be avoided by using standardized and widely used techniques like Transmission Control Protocol (TCP) and Hypertext Transfer Protocol (HTTP).

Because OPC UA is based on service-oriented architecture it gathers wide range of service sets. These services provide all OPC UA functionalities like read and write. Services are grouped into 10 service sets. Services are grouped by their purpose for example Method service set, Node manager service set and Attribute service set. [Virta 10, p.18]

Because of numerous classic OPC products out in the market, OPC UA has been developed to be backwards compatible with classic OPC. So, old products can be used with OPC UA products. There are already several wrappers which can be used to manage in this task. [Kepware]

Other reason for development of UA is the lack of information which can be added to signals with OPC. OPC UA brings capability to model complex information. Developers can add a lot more data into signals than in classic OPC. Information can be hierarchical so it also brings semantic information into models.

There also was a need for a real scalable technique. OPC UA can be used from embedded systems to Enterprise Resource Planning (ERP) systems. So it also covers some needs of Supervisory Control And Data Acquisition's (SCADA), Distributed Control System's (DCS) and Manufacturing Execution System's (MES).

Classic OPC doesn't take a stand on Security issues. Although, DCOM has some security features, they are hard to configure and often security is not set on because of this. Lack of the security is a pretty big issue in today's factories which can be maintained remotely. OPC UA has taken this seriously and added security into specification. A developer can now choose from few different setups of security depending on device capabilities. For example, embedded systems can use light security because of shortage of resources. [Mahnke 09, p.9]

OPC UA specification consists of 13 parts which are showed in Figure 2-6. In this thesis we are interested in part: 1, 3 and 5. Especially part 5 is important because it is about Information Model.

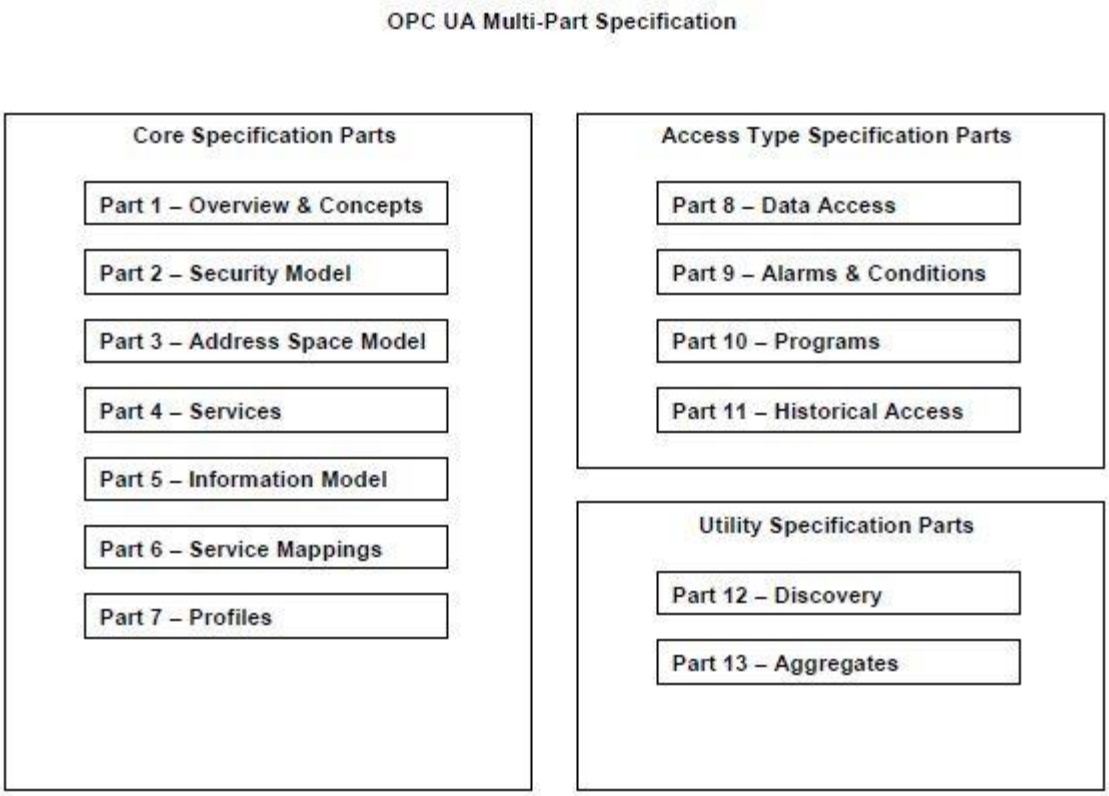


Figure 2-6: OPC UA Specification parts [OPC UA.1 12, p.6]

Standardized structure of the OPC UA server is presented in Figure 2-7. This is the basic structure of the OPC UA server where all information is stored. However, this is not strictly defined in the specification because some smaller devices which don't have that much memory can make own kind of structure if needed. Structure is constructed from *Folder Type* objects. All Types are in the *Types* folder, also self-made types. All instances are in the *Objects* folder. Server instance hierarchy is constructed under this folder.

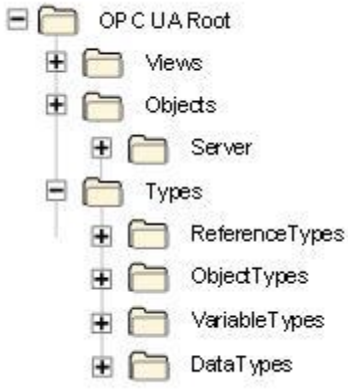


Figure 2-7: OPC UA Base root hierarchy [OPC UA.5 12, p.49]

OPC UA enables connecting different OPC UA servers seamlessly together. Every object is in some namespace and every namespace should have a unique name and Uniform Resource Identifier (URI). Using this URI, data from different servers can be linked by using suitable *Reference Type*. This technique allows designer to make multi-

level server chains, like in Figure 2-8. For example low level devices like valves or PLCs can have a simple UA server in them and upper level servers can connect with them directly using references. Some servers serve also as clients in these servers in chains.

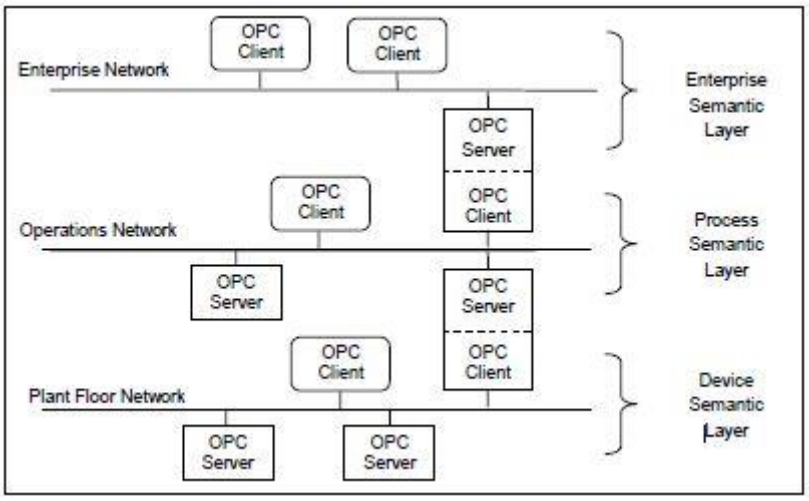


Figure 2-8: Chained OPC UA server example [OPC UA.1 12, p.16]

OPC UA uses its own software layers. These are OPC UA stack, OPC UA Client Software Development Kit (SDK) and Server SDK and Application layers. All these layers are in the Software layer if we think about Open Systems Interconnection (OSI) model. The software structure for servers and clients is presented in Figure 2-9. Figure also shows how all these layers can be done with different platforms.

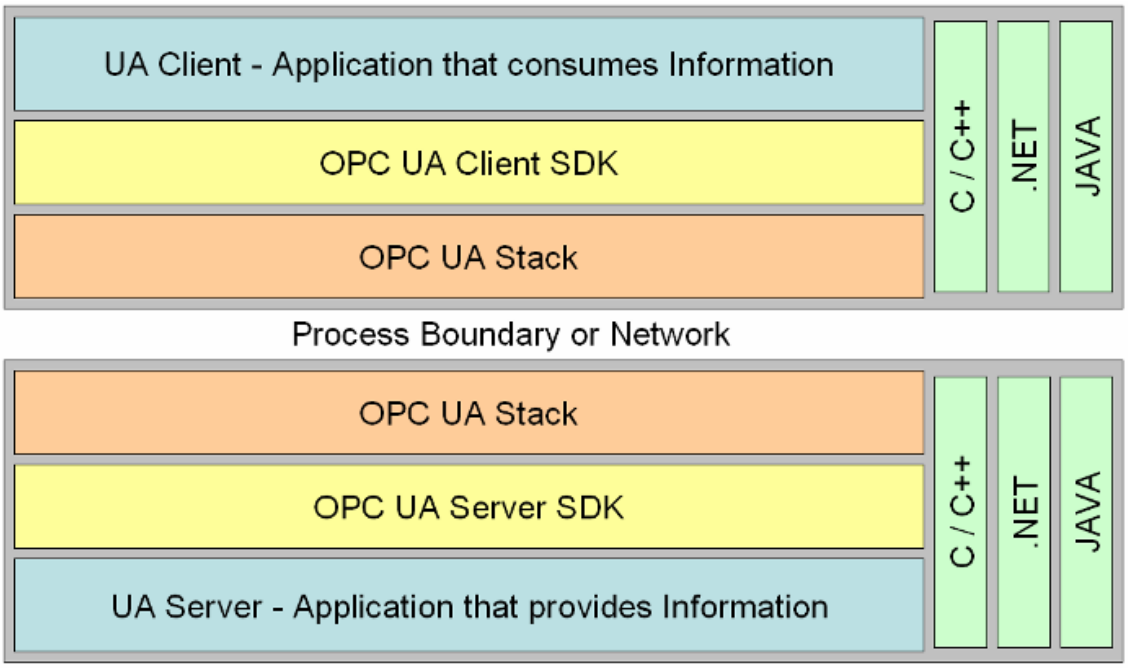


Figure 2-9: OPC UA software layers [Mahnke et al. 09, p.14]

SDK includes all necessary and frequently used features that the server or client designer might need. For example write and read features.

All data in the OPC UA server is stored in nodes. Nodes have several attributes specified by the OPC UA. Nodes can also have user added variables, objects and methods. Other main building blocks in the OPC UA namespace are references. References can't hold any data, but it can offer semantics by its type. References can be hierarchical or non-hierarchical. Hierarchical types are used for hierarchical purposes like between an object and its variable. Reference may also have a direction if it is non-symmetrical, like a parent child relation. In the other way the relation is "has-parent" and the other way it is "is-child-of". Even a reference itself is not a node its reference type is. This way references can hold certain semantics. Every node has a Browse name attribute which is the name of the node. The reference type can also have another name which is called Inverse Name. This attribute is used with non-symmetric references to describe reference's inverse semantics. [Mahnke et al. 09, p.22]

2.3.1 Information models

Generally speaking information models are sets of readymade types of objects, variables, references, events and data. First level information model above OPC UA base information model includes new types which are inherited from basic types in the OPC UA base information model. Next level information models can use these new types or base information model types.

There are readymade general information models which are called information model specifications or companion specifications. At the time of writing this thesis there are five specifications which are Device Integration (DI), Analyzer Device Integration (ADI), Field Device Integration (FDI), PLC Open and ISA-95. PLC Open, FDI and ADI types are partly inherited from DI model and other types from the base model. Development for few other specifications has been started. To list a few, there are Electronic Device Description Language (EDDL), Field Device Tool and AutomationML. In this thesis we are interested in ISA-95, DI and ADI models. General model hierarchy is presented in Figure 2-10. [Mahnke et al. 09, p.11]

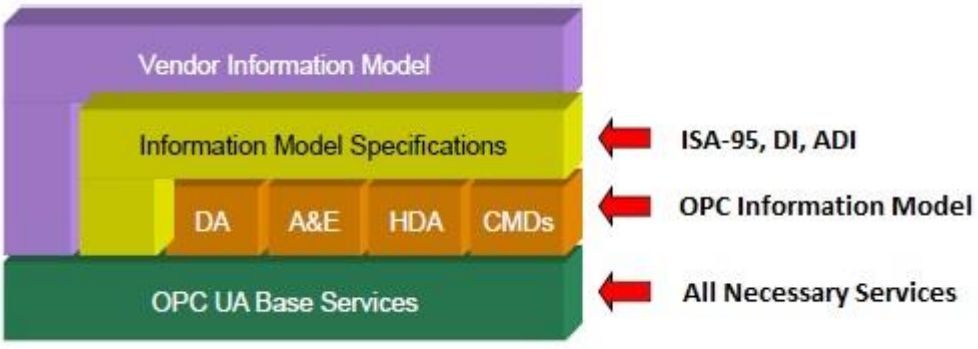


Figure 2-10: OPC UA general information model hierarchy

When making new Types designer can use Modelling rules to specify what Type properties are optional and what are not. Modelling rule options are Optional, Mandatory, Optional Placeholder and Mandatory Placeholder. With these rules designer can choose what variables instances must have and what are optional. This can also be used with objects and methods. Properties which have placeholder modelling rule are like list of properties.

It is also possible to use subtyping which is also known as override. It means that modeler can extend existing Types. Sub Types can extend or restrict some features of the parent Type. For example if parent Type have a Variable whose data Type is numeric, sub type can restrict the data type to float but not to string.

2.3.1.1 OPC UA information model

OPC UA information model includes lots of general types which usually are needed in most of the UA servers. These types are for example data types like integer and string or reference type like 'has child'. This way all OPC UA servers have the same basic structure. Information Model defines the Address Space of an empty UA server. This doesn't mean, however, that all servers must provide all these basic types. OPC UA uses inheritance when defining types. For example all data types are inherited from a base data type, similarly events, objects, references and variables have their own base types. Some of these base types are presented in Figure 2-11. [OPC UA.5 12, p.1]

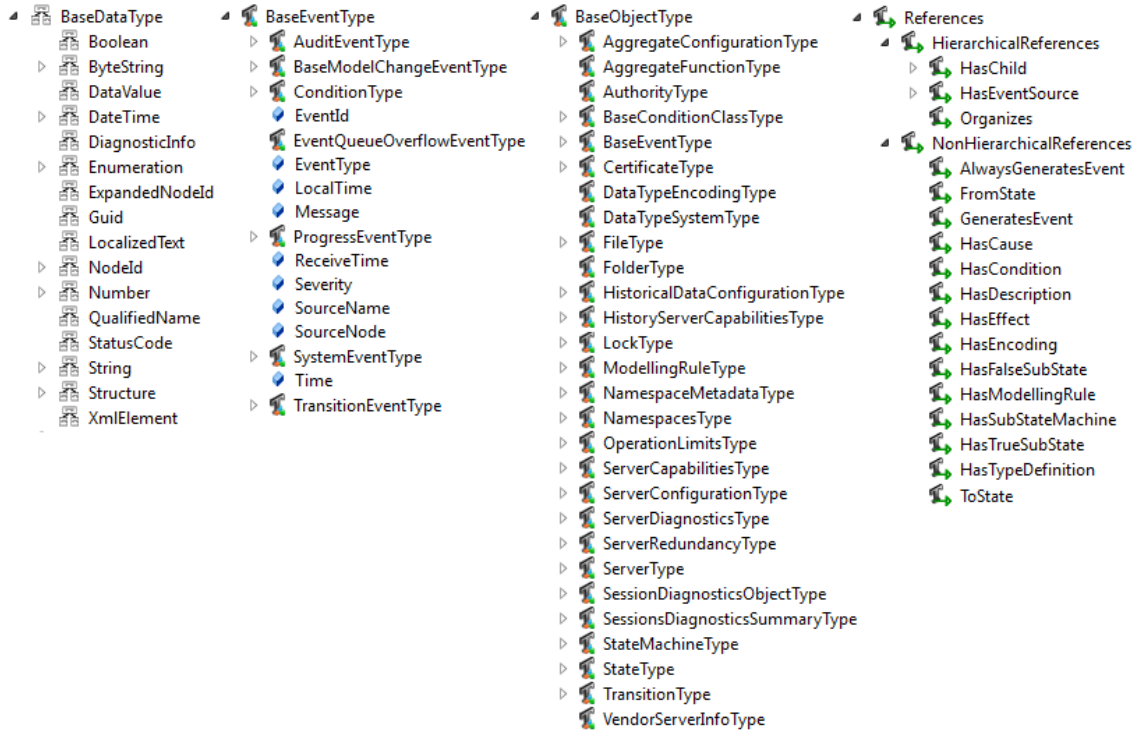


Figure 2-11: OPC UA base types from UaModeller

Symbols used in Figure 2-11, are just UaModeler own icons. All data types are for example grey symbols at the left side of the figure. Blue cubes in the base event type pre-

sent base event type properties. If there is little triangle at the right side of the type, that type has subtypes.

Model specifications can be imported into several tools. OPC Foundation has standardized XML format for these files and it is called UaNodeSet [OPC UA.6 12, p. 68]. There is also another format which is used by some tools and it is called ModelDesign. ModelDesign format is used by OPC Foundation's ModelCompiler tool for example. This tool will become obsolete and after that ModelDesign format also. [Laukkanen 13, p. 21]

2.3.1.2 ISA-95 information model in OPC UA

ISA-95 OPC UA model was developed in order to facilitate and standardize ISA-95 implementations in OPC UA servers. This helps to make OPC UA servers that implement ISA-95 specification. This helps them to be more compatible with other ISA-95 products in the future. The ISA-95 OPC UA information model is focusing on a common object model. This model includes five resource types which all company specific types should be inherited from. These types are Person, Equipment, Physical Asset, Material Sub Lot and Material Lot. There are also different kinds of test specifications for each of the types mentioned earlier. ISA-95 also defines new data types and reference types which are in the ISA-95 Base information model. This structure is presented in Figure 2-12.

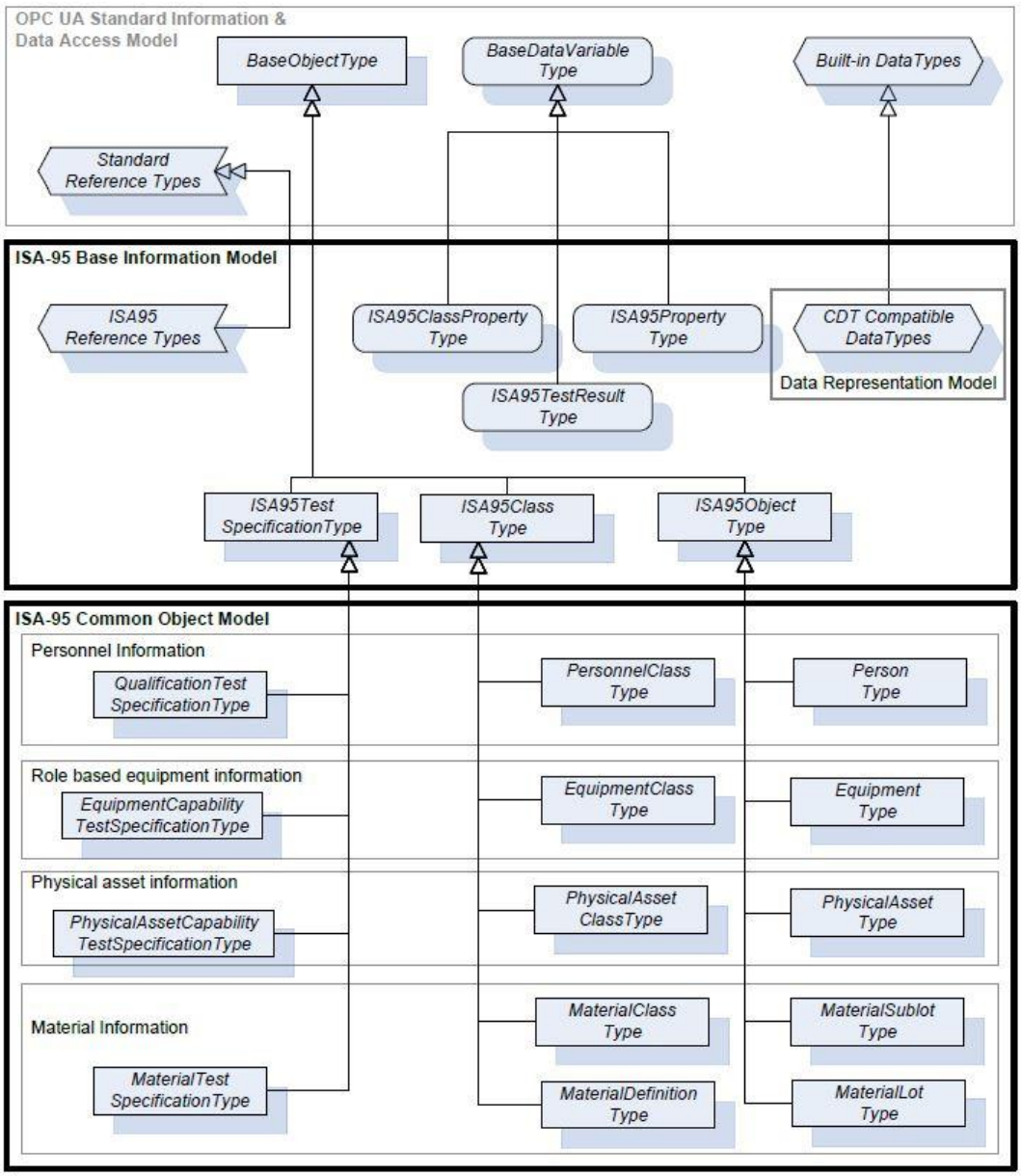


Figure 2-12: Structure of ISA-95 models in OPC UA [OPC UA ISA-95 13, p.27]

More models will be introduced in the next release of companion specification. These are Logical View of resources and Production Activity models. They are presented in Figure 2-13. [OPC UA ISA-95 13, p.6]

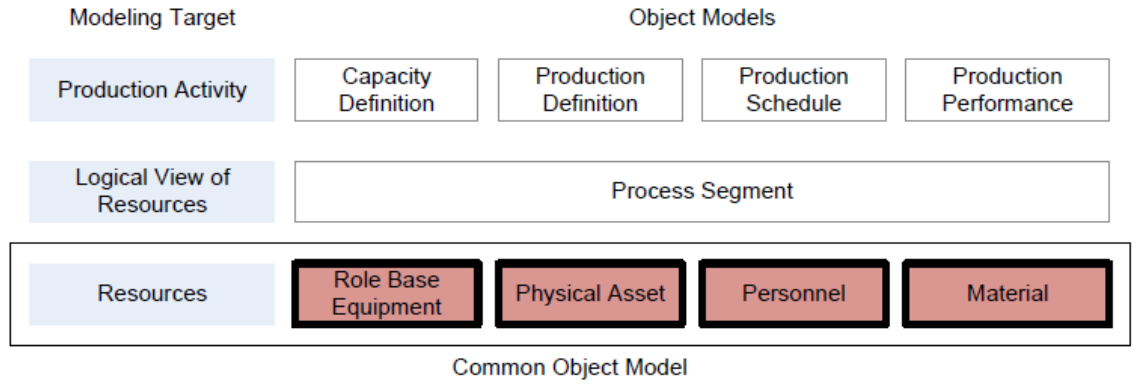


Figure 2-13: ISA-95 overview [OPC UA ISA-95 13, p.6]

The ISA-95 offers a standardized structure of information for OPC UA. Here ISA-95 creates a structure of the data and OPC UA offers ways to transport the data.

The ISA-95 itself is an abstract specification, so it doesn't have data model. Business to manufacturing Markup language (B2MML) implementation adds data types from UN/CEFACT Core Data Type (CDT) specification. Some data types are already in the OPC UA specification and rest of them are added. [OPC UA ISA-95 13, p.28]

When the ISA-95 styled equipment model is modelled using OPC UA, it can be done in a very flexible way. Some ISA-95 specific tools are quite strict about the structure of the equipment hierarchy but the OPC UA modelling style gives a very good foundation for agile modelling. Types in OPC UA are easy to create and instance hierarchy can be done as deep as a designer wants. Although ISA-95 doesn't specify more than 5 levels for equipment like in Figure 2-3, designer can add additional levels under work unit levels making the hierarchy deeper. Other option is to use levels from other specifications like levels from ISA-88. The ISA-88 is an extension of the ISA-95 and was initially designed for a batch process only.

2.3.1.3 Device Integration (DI) and Analyzer Device Integration (ADI) information models

Device integration model also known as devices companion specification is one of the first companion specifications out there. It is foundation for many other companion specifications like ADI and PLCOpen. This kind of model hierarchy is presented in Figure 2-14. Its purpose is to decrease integration effort of different devices from various manufacturers. It contains three different models which are device model, device communication model and device integration host model. It is intended to be a foundation for device manufacturers who want produce devices that offer OPC UA interface. Its cornerstone is *topology element type*. It is used for making device topologies. Topology elements have *parameter set* and *methods set*. Other main object is a *functional group type* which is used for organizing these parameters and methods. Last main building block is a *device type* which contains lots of optional data like *serial number*, *manufacturer* and *model*. It is inherited from the *topology element type*. Another kind of type is a *protocol type* which can be used with communication protocols like Profibus. [OPC UA DI 13, p.7]

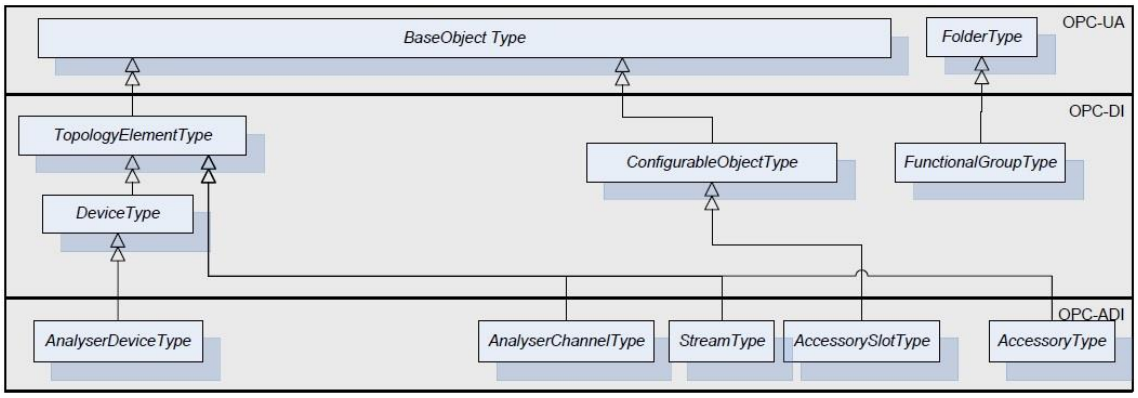


Figure 2-14: Part of OPC UA Analyzer Device Interface model [OPC UA DI 13, p.8]

Analyzer device companion specification is designed for analyzer devices only. It extends many types which are defined in the DI model. Some of these are shown in Figure 2-14.

An analyzer, in general, consists of many channels which can analyze an analyte. One channel may contain several streams where it can take the analyte from. Data is in a raw format after analyzing. This data is used with an analyzer model to generate a scaled data. The analyzer model is a mathematic model. The analyzer model uses configuration parameters which user can modify. The scaled data can be used with a chemometric model which is also a mathematic model and it generates one or more process values. This data flow is presented in Figure 2-15.

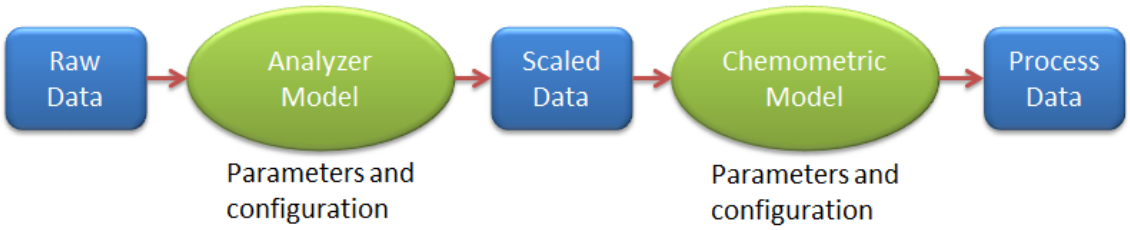


Figure 2-15: Mathematic models of the analyzer

The ADI model consists of all these parts and features which can be used to build one specific analyzer model. There are several different types of analyzers for example particle size monitoring system, light spectrometers and chromatographs. All of these types have their own base models defined in the ADI specification. In this thesis we are interested in a particle size monitor model. These specific models contain parameters and methods which are common for them. [OPC UA ADI 13, p.2-5]

Analyzers which provide an ADI interface can be used remotely. One use case can be for example when parameters for a mathematic model need to be updated. A maintenance person can update parameters using a remote interface through the OPC UA. The ADI model is not used to gather raw data from the process. Instead it is designed to gather process values which are calculated using mathematic models.

Analyzer consists of several different types which are, *Analyzer Device*, *Analyzer Channel*, *Stream*, *Accessory* and *Accessory Slot*. General structure of an analyzer is presented in Figure 2-16.

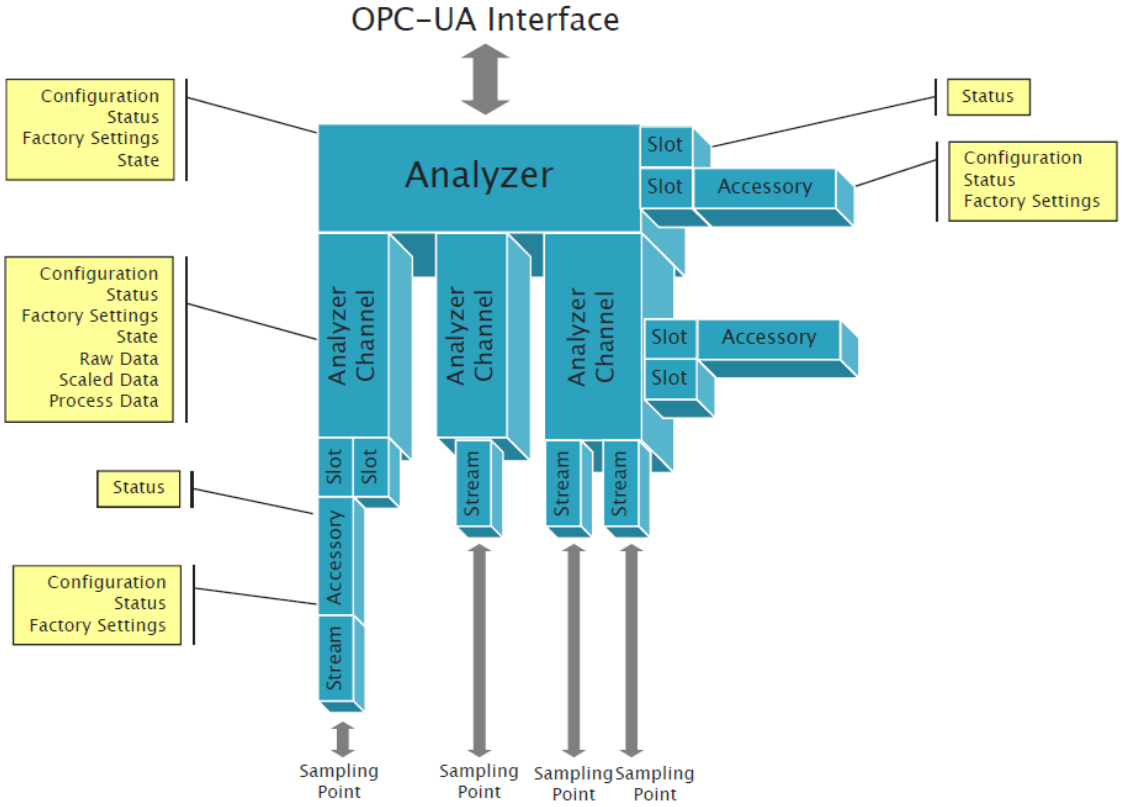


Figure 2-16: General structure of the analyzer [Brandl 09, p.33]

Often one analyzer can have more than one analyzer channel. The ADI model supports this feature. There is also a possibility to connect accessories into the analyzer or into the analyzer channel. The analyzer can process more than one analyzer channel at a time but only one stream can be used in one channel at a time. The ADI specification doesn't specify all parameters for analyzers but instead collects general set of parameters which are commonly used in every analyzer. These are for example *configuration*, *status* and *factory settings*. They are a collection of many parameters. There are also several different methods like *set configuration* and *reset all channels*. Analyzers are state based machines and the ADI model has taken care of this by adding a state machine into the analyzer type. Every channel has also own parameters and methods. Parameters are for example *Is enabled*, *channel id* and *diagnostic status*. Channel methods are for example *Go To Maintenance* and *Start*. Channels also have their own state machines. Every stream may also have own parameters. Parameter groups for streams can be for example *chemometric model settings*, *acquisition data* and *status*. *Accessory slot* can be used to limit supported *accessory types* which can be connected to an analyzer or a channel. Accessories are for example auger, filter wheel or detector. [OPC UA ADI 13, p. 9-40]

2.4 Unified Modelling Language (UML)

Unified modelling language (UML) diagrams are used in this thesis in several different cases. It will be introduced briefly. UML consists of several different diagram types, like class, use case, activity and sequence [Russ et al. 06, p.11]. In this thesis and also in the ISA-95 specification the most used diagrams are class and object diagrams. In this chapter we focus on these two diagrams to give a better understanding needed later on this chapter.

UML class and object diagrams are generally used in software design, but can also be used in other situations. UML rely on Object-Oriented Encapsulation (OOE). Objects and classes can be thought like in the next example, which explains OOE. Class is like a sand cake mold and object will be the sand cake itself. So class is like a list of instructions how to make objects. If class have a property x then an object of that class also have its own property x which may have some value. Objects and classes can have properties and methods. Figure 2-17 shows this. *Instance Type* is just a general class with example properties and how class diagrams are presented in the UML class diagram. UnitType object have a public property *Name*, a private property *ID* and a public method *work*. Figure 2-17 only presents things we are using in this thesis. UML class and object diagrams include a lot more features than this. [OMG 2011, p.1]

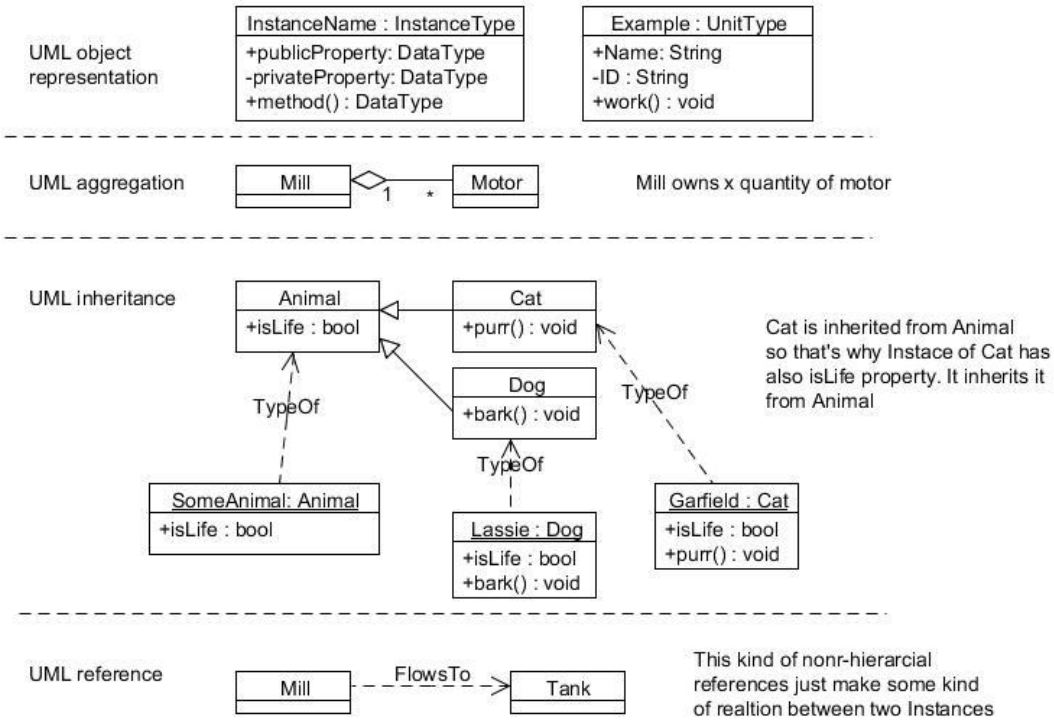


Figure 2-17: Generally used UML notation in this thesis in class diagrams

Next in Figure 2-17 there is an example how to tell that some object belongs to another object. It's called aggregation and it is presented with an empty diamond. The number one and the star mean that one *Mill* has zero or many *Motors*.

Next part in Figure 2-17 presents inheritance. *Animal*, *Cat* and *Dog* are classes and *Some Animal*, *Lassie* and *Garfield* are objects of those classes. *Type Of* arrow just presents that objects and classes can have specific relationships. Here the *Cat* is inherited from the *Animal* so all objects which are of type *Cat* has *is Life* property from the *Animal* and *purr*() method from the *Cat*. Inheritance is presented with an empty triangle headed arrow. [Russ et al. 06, p. 63-100]

2.5 Mapping between UML notation and OPC UA notation

OPC UA specification is not using UML diagrams. This chapter explains how OPC UA markings are mapped corresponding to the UML notations. This helps to understand some figures in this thesis and it is easier to compare different figures if they are drawn with different notations.

Figure 2-18 presents how OPC UA types and instances correspond to UML classes and objects. OPC UA types are like UML classes and OPC UA instances are like UML objects. The term 'object' is often used instead of an 'instance' in the OPC UA specification as well. In the UML when some class owns another class, diamonds are used on the parent side to describe that it owns the other object. Figure 2-18 shows an example of this, where *Motor Type* owns *Config Type*. This relation is called composition. In OPC UA there is *Has component* arrow to present the same thing. Type for *Configuration* is not defined in this figure for OPC UA but it is modelled same style as *Motor Type*.

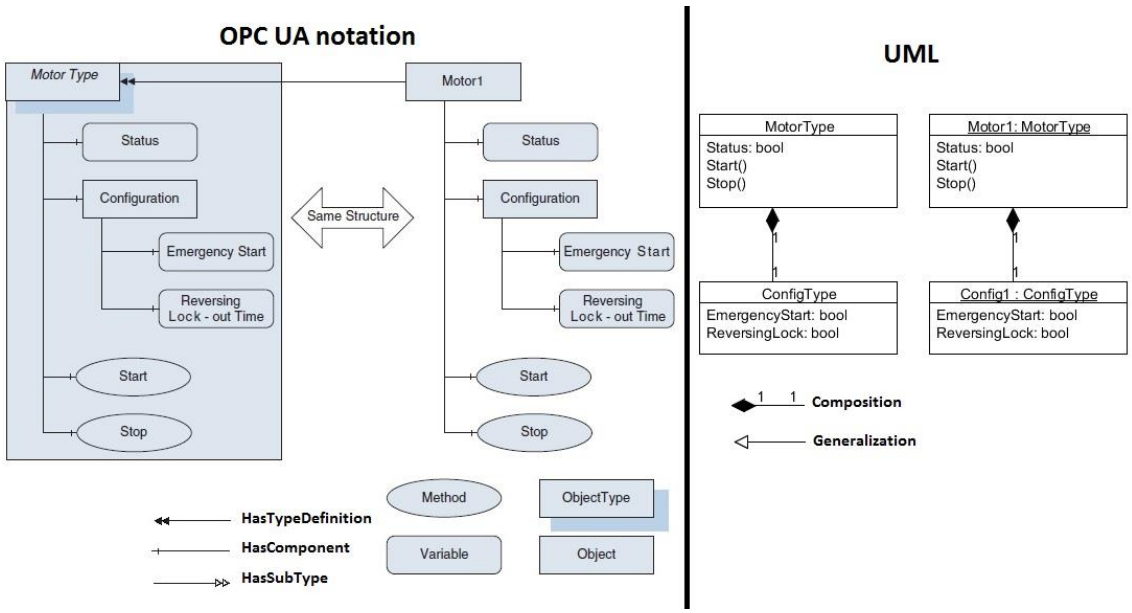


Figure 2-18: OPC UA notation versus UML

Has sub type and *generalization* are not used in this example but they correspond to each other. *Has type definition* reference has some kind of equivalence with UML class diagrams, where object name and object type are underlined. Like in right side of Figure

2-18 there is *Motor1* which is an object of the *Motor type* class. Underlining in UML means that it is an object not a class.

Both of these specifications have more features than explained here. Only parts relevant to this thesis were covered here. There are many OPC UA notations that don't have clear equivalents in the UML. For example there are no equivalent notations for views.

2.6 Computer Aided Engineering Exchange (CAEX)

The core idea behind CAEX is to automate data transportations between different tools. Many mechanical designing tools have this feature, but not all. Goal is to reduce duplicated work of effort.

The term CAEX comes from Computer Aided Engineering Exchange. It was developed at first for process engineering. It is based on an XML format. Plant data and structure can be modelled with it. It is a part of an AutomationML standard group. CAEX base structure consists of four libraries which are *System Unit Class Library*, *Role Class Library*, *System Hierarchy* and *Interface Class Library*. These libraries include structures and tools for modelling plants and their functions. A plant hierarchy is constructed into System Hierarchy using System Classes, Role Classes and Interface Classes. Example structure of this is presented in Figure 2-19. In every library there are *Resources*, *Products* and *Processes*. This is called Process, Products and Resources (PPR) model. [Schleipen 10, p.506]

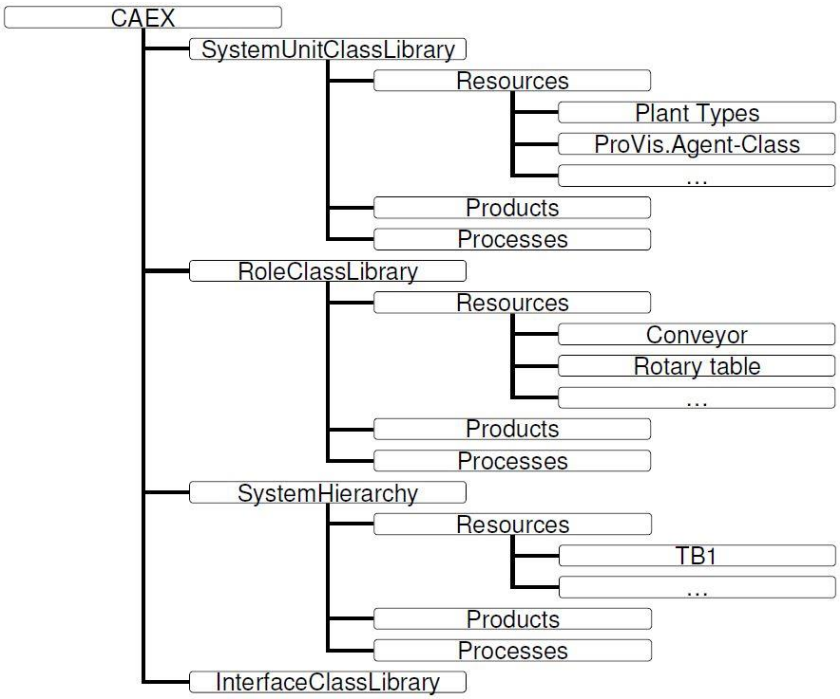


Figure 2-19: CAEX-PPR-model [Schleipen et. al. 08, p.1789]

CAEX is introduced in this thesis because an alternative for the ISA-95. Both can be used for modelling plants and their structure, but the scenario is different. There is a working group for developing a CAEX based OPC UA model specification and it is introduced in the next chapter. ISA-95 and CAEX models are compared later.

2.7 OPC UA and CAEX

A benefit of combining OPC UA and CAEX is to get the best features of both. OPC UA and CAEX have their own certain roles. OPC UA makes platform independent and standardized data transmissions possible, whereas CAEX makes standardized data format possible. OPC UA is answering to a question: "How does communication work?", and CAEX is an answer to a question: "What is communicated?"[Schleipen 10, p.510].

How CAEX and UA are used together is explained next. CAEX data is generated for example from several different design tools. Based on that data OPC UA objects and properties are generated by OPC UA client which is designed especially for this use. When a new item is created by the design tool which supports CAEX that item will be added automatically to the OPC UA server. Another OPC UA clients will be notified that a new item was added. This concept is presented in Figure 2-20.

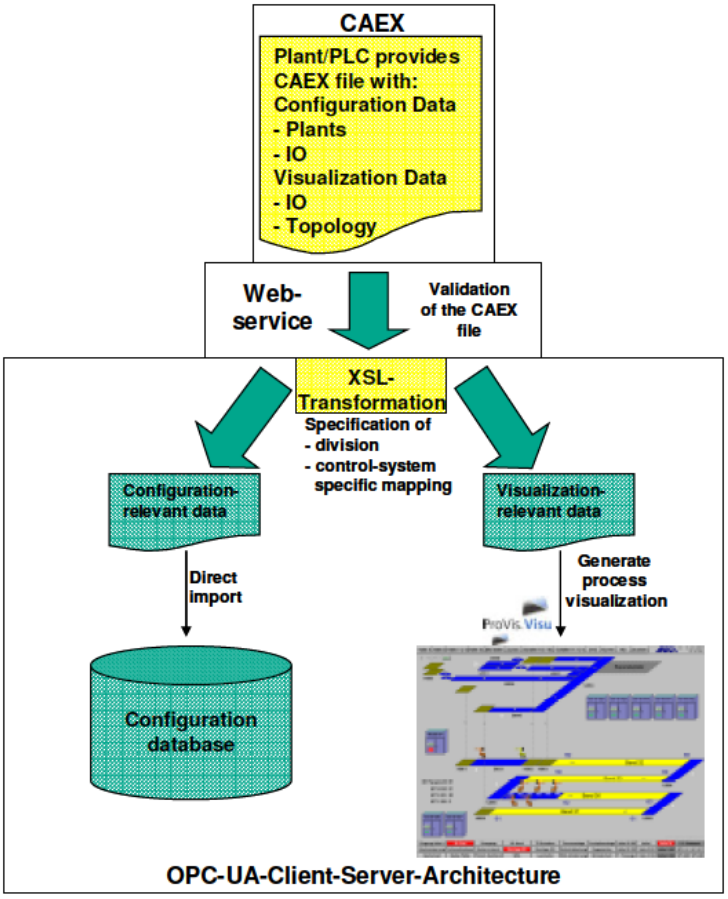


Figure 2-20: The engineering framework [Schleipen et. al. 08, p.1790]

The AutomationML standard group is using CAEX for a top level data structure. OPC UA AutomationML model which is using CAEX format is still under construction by Fraunhofer IOSB at the time of writing the thesis. [Henßen 10, p.2]

Both these standards, OPC UA and AutomationML are based on a type-instance concept and that is why it is easy to start combining these two standards. The main goal is to make AutomationML companion specification which can reduce errors and increase reusability of objects and items. If models are made well, transformation from a CAEX file to OPC UA server objects can be done automatically. [Henßen 10, p.7]

2.8 Comparing ISA-95 and CAEX models

CAEX divides items in a plant by PPR model, whereas ISA-95 divides items to Equipment, Physical Asset, Material and Person. In some particular cases they both can be used in same situations but due to differences in division the result will be quite different.

The CAEX also has different styled object making process than in the ISA-95. In the CAEX, objects are made by System Unit Class, Role Class and Interface Class together. Object structure in the ISA-95 is more of the same style as in the OPC UA. In the ISA-95 objects are more like objects in Object-Oriented Encapsulation (OOE). In OOE objects have properties and methods, of which some may be private and other public or something in between. But in the ISA-95 objects have only properties.

The ISA-95 is made to be an abstract foundation for managing functions and operations of the plant, whereas the CAEX is made for plant's physical modelling. However, if we want to compare objects divisions we can notice that it is almost pointless because both meta models are so far from each other. But if some relation between these must be made, Equipment and Physical Asset are related to Process and Resources, Materials and Persons are related to Resources. In the ISA-95 there is no relation to products. All these relations are quite loose. Building blocks of the objects also force that there is no way for example to convert CAEX model to ISA-95 model.

Usages of these are also quite different from each other. The CAEX was first designed for neutral data format for different engineering tools. Now there is a project on-going where CAEX model is tried to be combined with the OPC UA. When they succeed in this it is pretty simple to convert CAEX formatted data to the OPC UA using a ready-made converter. The CAEX model can also be used with the OPC UA design tools, but then CAEX idea for generating an OPC UA server directly from the CAEX formatted data, disappears.

ISA-95 model is already done for OPC UA, but there are not yet many tools which use it. ISA-95 model specification can be used by several different tools which are designed for OPC UA. An engineer can for example design ISA-95 objects with any OPC UA design tool which has OPC UA XML import feature implemented. In this case an engineer can import the ISA-95 model from the XML file into the design tool and extend ISA -95 objects by inheriting them from ISA-95 objects. For maintaining an ISA-95 based OPC UA server, engineers can use for example Easy95 software which is designed for this use.

The ISA-95 was first designed for the use of MES and ERP. OPC UA ISA-95 server will also become tool for the MES and the ERP systems. Here OPC UA is a tool for connecting MES systems to lover level systems like PLCs and DCSs. At the same time the ISA-95 model makes integration easier for MES level systems.

Software which use a CAEX formatted UA server are more likely SCADA software where User Interface (UI) design plays a bigger role. Many CAEX examples include UI components. The OPC UA CAEX model will hold this kind of data also. One goal of CAEX is to standardize UI components. Other goal is to hold the information which is needed to connect the OPC UA server to lover level systems like PLC.

3 GENERAL MODELLING PROCESS

This chapter introduces how to build models using different tools. At first modelling process will be introduced. This includes several steps which should be gone through. After that there are several different aspects which should be considered when designing with ISA-95 and OPC UA models. Designing tools were found for this thesis just by searching from the internet and asking from OPC UA professionals. All the tools that were found will be researched in this thesis.

3.1 Design

At first designer should research what kind of devices there are in a modelled plant. Next step is to figure out what signals and data can be received from a device and what the upper level systems requires. This kind of data can be found for example from Piping and instrumentation diagram (PI) diagrams, loop lists and by interviewing engineers which are familiar with the plant. These steps are presented in Figure 3-1. These data gathering steps and also modelling steps was validated in meetings with the engineers who are familiar with the example plants.

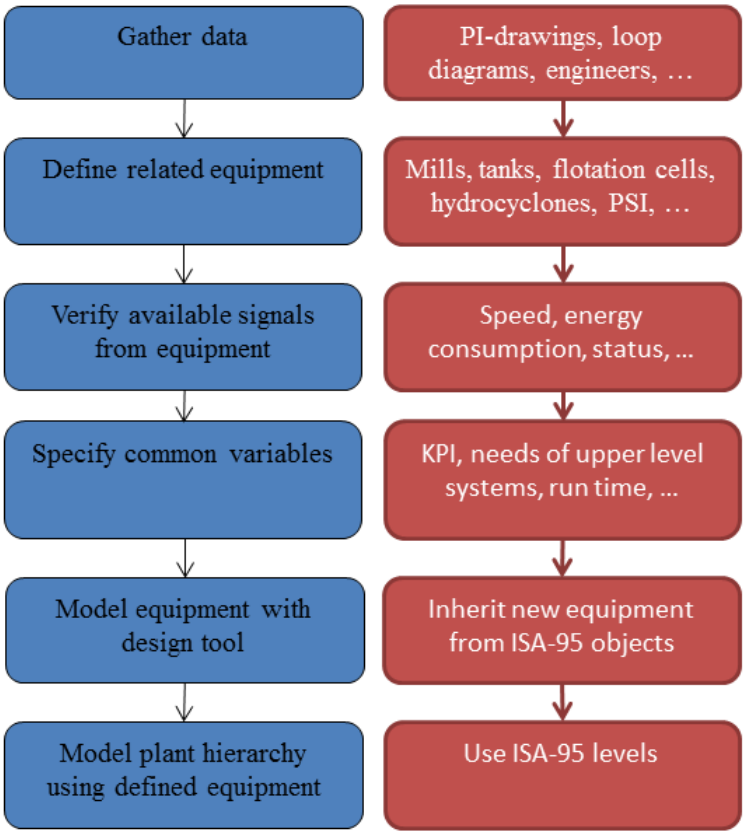


Figure 3-1: Plant modelling phases

Next step is to choose a right modelling tool. Other thing is to decide how to maintain server address space. There are some tools available for both of these things but one option is to make own custom software for this, at least for maintenance. There are already good modelling tools for designing models. These tools are tested in the next subchapter.

In practice designer first installs a chosen design tool, for example UaModeler. Designer starts to expand objects by inheriting readymade Types and making new Types. These Types are modeled devices like a mill or a tank. Designer can also test how to model instances with these tools but usually these tools are not meant for this kind of work. When needed devices are modelled model can be generated to a real OPC UA server which includes these device models. Code generation is implemented in many modelling tools. Designer can add own features into the generated code before starting the server. Clients can start to build an instance space into the server address space when server is running. At the same time other clients can start to use these instances and data.

When modelling ISA -95 styled models *ISA-95 Class Type, ISA-95 Object Type* and *ISA-95 Test Specification Type* and their inherited subtypes should be extended. These types were introduced in Figure 2-12. Same concept can be utilized for other ISA -95 Types. For example when modelling some device, *Equipment Type* should be extended. That new Type can now have some specific properties that the device has, like *speed* or *energy consumption*. It is also possible to add more information for every signal if signals are for example *Analog Item Type*. *Analog Item Type* includes several optional variables which are usually linked to signals, like *engineering unit*, *ranges* and *definition*. Before designer starts to make new types he/she should check if a suitable type already exists in the OPC UA or in the used companion specification.

There is a resource classification definition for every ISA-95 resource model. In this thesis resource classification was left out of the scope. For Equipment there was no need for Equipment Class. Here class doesn't mean OOE styled class. It is rather a classification. One possible way to use Equipment Class is with Equipment with certain role like safety or emergency. Equipment Class should be used if one can find clear way to utilize it. In this thesis we only use Equipment and Equipment property for modelling.

When modelling these new Types for the equipment, they should be modelled in a separated namespace. This makes them easier to take into use in other projects. This kind of design is presented in the Figure 3-2. Figure presents that ISA-95 Types are inherited from the OPC UA Standard Information model Types like the ISA-95 Object Type is inherited from the Base Object Type which is located in the Object Types folder. The next step is to inherit Outotec specific Types from the ISA-95 Types. The Outotec specific type model is a plant specific model for example one for the concentrator and one

for the tankhouse. These type models can be used for several same styled processes. Therefore every plant uses the same types. The final step is to make an Outotec specific instance model for every plant.

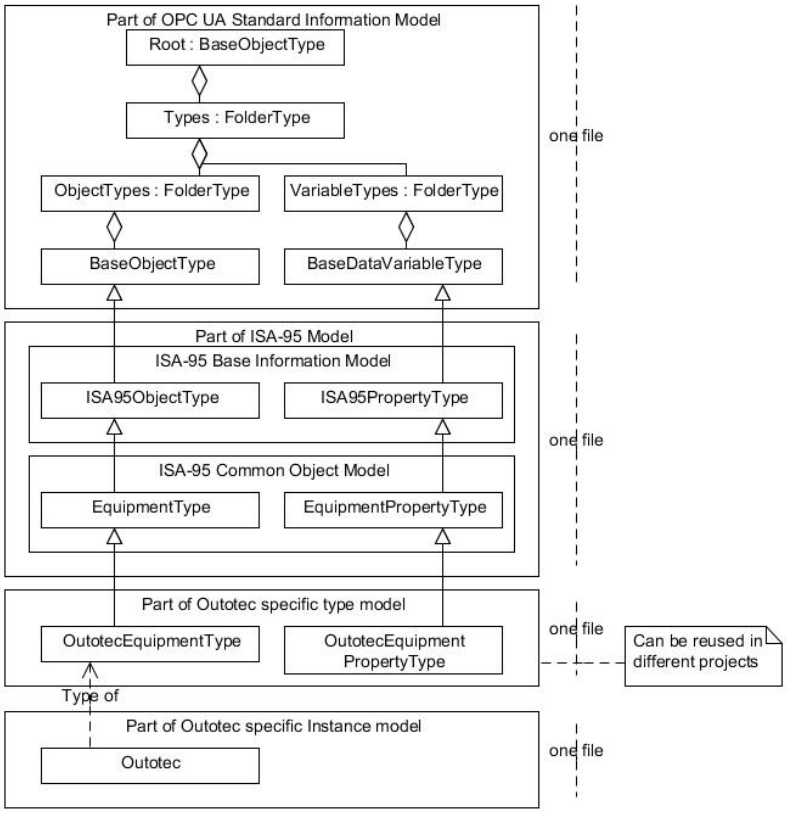


Figure 3-2: Outotec OPC UA model hierarchy

CAEX model for OPC UA doesn't exist yet so development with it can't be done. But when it is released, server creation from CAEX formatted data should be automated with some tool. Then CAEX can be used efficiently. For example, modelling would be proceed so that first CAEX data from engineering tool which support CAEX is exported, then some converter that creates an OPC UA server using that CAEX data is used. Other version might be that an OPC UA server is already created and when new CAEX item appears into a CAEX database it triggers code which generates new item into the OPC UA address space.

3.2 OPC UA modelling tools

Chosen modelling tools are UaModeler from Unified Automation, OPC UA Address Space Model Designer from CommServer and OPC UA Modeler from Fransuhofer IOSB. With these tools designer can design an OPC UA address space. Designer should make only device types with these tools not instances. This is because these tools are not designed to be maintenance tools for OPC UA objects instances. There are already some tools for maintaining address spaces, for example for ISA-95, there is Easy95,

which is an OPC UA maintenance tool designed especially for ISA-95. Features of different tools are presented in Table 3-1.

Table 3-1: OPC UA modelling tool features

Feature\Tool	Unified Automation: UaModeler	CAS: OPC UA ASMD	Fraunhofer IOSB: OPC UA modeler
XML import	X	X	X
XML export	X	X	X
CAEX support			X
Code generator	X	X	
Hierarchical view	X	soon	
Graphical view	X	X	X
Data binding		X	
Model format	UaNodeSet	ModelDesign	UaNodeSet
Version	1.3.3	3.00.02	-

XML import means that tools are capable of importing companion specification models or other models from an XML formatted file. XML export means that tool can create new XML formatted files which describe new models. CAEX support means that it can convert CAEX formatted data into an OPC UA address space. Code generation means that tool can generate code which can be compiled into an OPC UA server. Hierarchical view is useful when adding new types and instances into a namespace. Graphical view is useful if designer need to see several references of different nodes with graphical notation. It is only used for visual purposes. All development should be done with a normal view. CAS's modeler includes data binding feature which means that data can be linked to nodes in the design phase. Model format tells what XML format is supported for importing and exporting models.

3.2.1 Unified Automation UaModeler

UaModeler was in a leading role when modelling was tested with different companion specifications. UaModeler was the most advanced modelling tool from the selected ones. Further development of the tool has been continued and the latest release is from 6th May 2014 when writing this. That latest release includes few bug fixes that were found when testing this tool during this thesis work. This newest version still has some bugs in it, because it crashes when modifying certain things. One bug what was fixed happened when some instance was moved from one namespace to another. Another bug that was fixed happened when an instance was moved under another instance. The developer said that these crashes were caused by a complex model which ISA-95 models have. After the fixes UaModeler can take care of these complex models.

User interface which is presented in Figure 3-3 was consistent to use. User interface has two modes which are View and Edit which are respectively a graphical view and a normal view. View mode was left for minor use because it was hard to use since tool optimizes certain amount of nodes to be shown at the time. So if developer wants to open some node at the same time some other nodes are closed.

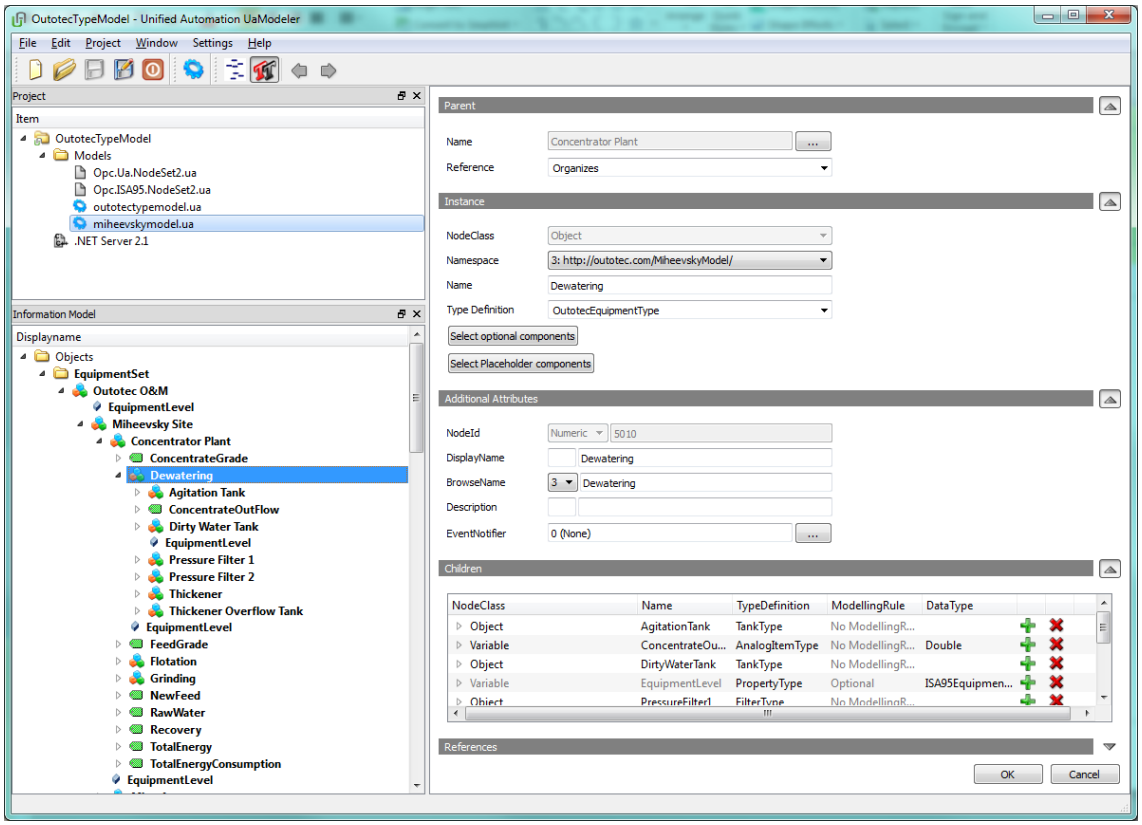


Figure 3-3: Unified Automation's UaModeler's user interface

The most important views of the edit mode are information model pane and project pane which can be seen left side in the Figure 3-3. Edit mode pane is on the right side of the Figure 3-3. Output, attributes and view panes can also be used. Developer can see models and a selected code generator at the project pane. New models or already existing models from an XML file or UaModeler's inner format file can be added from the project pane. UaModeler uses .ua file extension for the inner model format. Developer can browse all added models from Information model pane. Information model pane looks similar to most of the generic OPC UA client tools. It has the same structure as the generated server will have. This helps development a lot because it is easy to understand relation between different nodes. All instances are under Object folder and all types are under Types folder. Bold font is used for all nodes that aren't OPC UA's basic nodes. If developer needs to know what namespace node belongs to, developer can look it under the edit pane.

After models are done designer can generate codes which include classes for every new type. These codes can be modified afterwards for example with Visual Studio which is Microsoft's integrated development environment for developing computer programs.

What is also needed is to generate an XML file for every model. This thing can be done separately for each model. These files are loaded during OPC UA server's startup. This might cause some problems if a model is changed after the startup. Server should have a feature to regenerate these XML files for the new model. This might be an issue if server will shut down and models were updated by some clients when the server was on. During the next startup these modifications usually need to be on the server. In this case these new regenerated XML files should be loaded to the server.

If program does not need any modifications after code generation, program can be compiled and run. The generated server is executable software. These kinds of servers should usually be services that are always on. In this case the server can be encapsulated into a service.

References can be added for any node easily using the Edit pane. At the bottom of the Edit pane designer can add any reference to any node easily. This should be used only for other references than references used with property and instance references like *HasChild* and *Organizes* references. Features like adding properties and instances have their own mechanism designed specifically for them for example clicking right mouse button over instance and selecting add new instance.

3.2.2 CAS Modeller

This tool was left for less testing because the UI wasn't that sophisticated. The first notable thing was that every node at the left side was in a random order without any tree structure. However, in the menu there was a 'Sort the tree' feature that was only an advertisement for the new feature. Version that was tested was made in 2011 so these features may not even be under development since they have not been released after 3 years. There were also some other nice features that was not implemented yet. With these new features this could be a good competitor for UaModeler, now it is too incoherent to use when compared to UaModeler. User interface of the Address space model designer is presented in Figure 3-4.

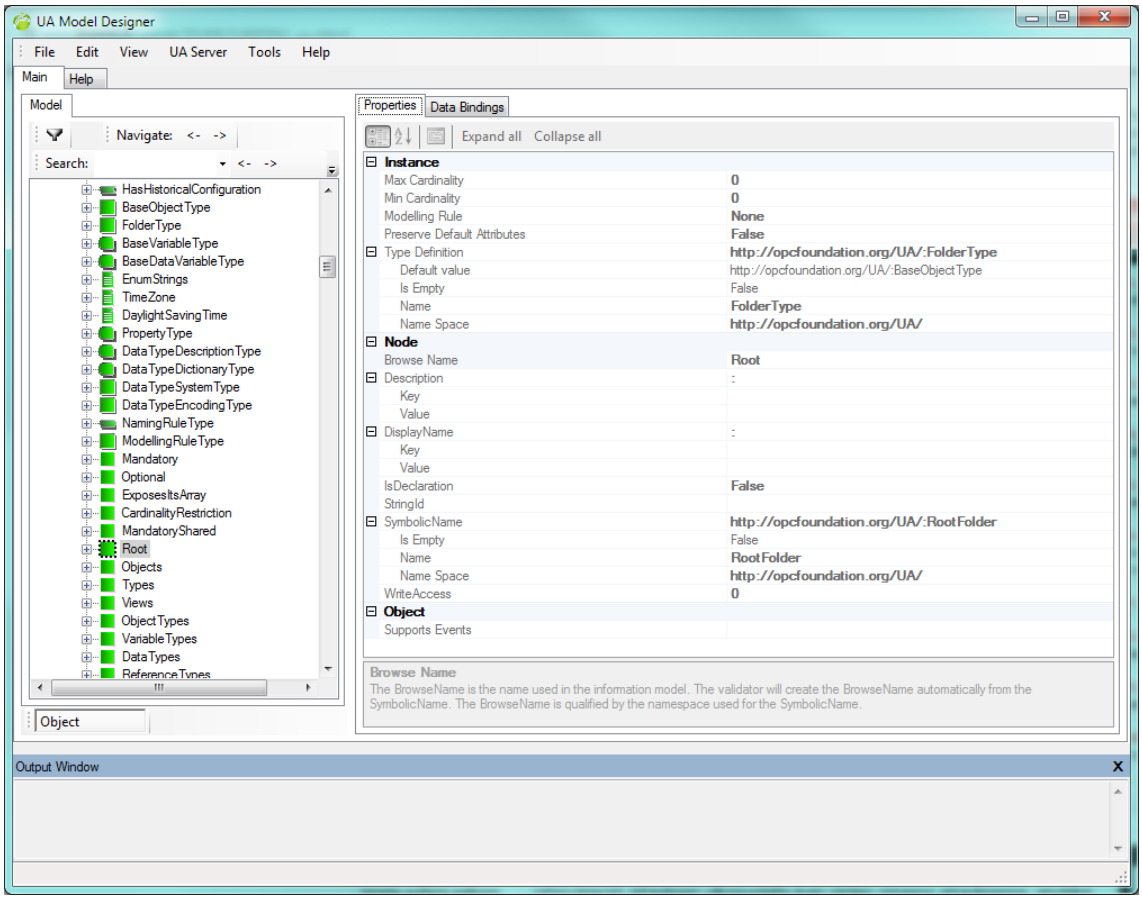


Figure 3-4: CAS's UA Address Space Model Designer

However CAS Modeller has one great tool for integrating an OPC UA server with the old OPC tools. With this tool designer can bind data directly to a model when modelling. Code generation was not tested with this tool, because the feature is only available for full version users only. CAS Modeler has also integrated some other tool like state machine editor. It could be used with ADI model's state machines.

3.2.3 Fraunhofer IOSBs OPC UA modeller

This tool was compared with the other two tools only with information received from the producer's internet site and from one developer from Fraunhofer. This tool has one different feature that other tools haven't. It has a CAEX import feature. This feature can be used for generating OPC UA servers from CAEX formatted data. The producer does not share any demo version of this tool so testing was excluded from this thesis. However, some more accurate information was received from one developer from Fraunhofer. Fraunhofer can provide workshops and consulting but not demo tools like from other two companies. However, they can offer a tailored tool for needs of one company. Features which can be selected are for example language, XML import and AutomationML/CAEX import. Figure 3-5 presents example user interface of Frauhofer's OPC UA modeller.

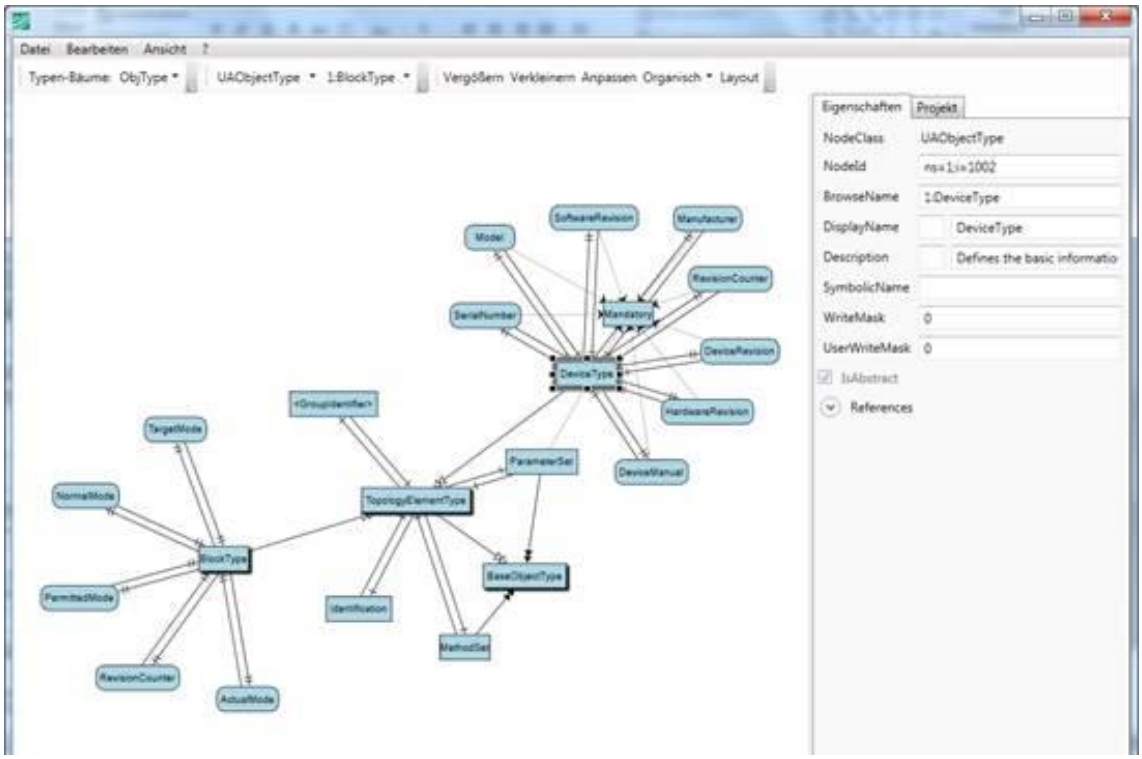


Figure 3-5: Fraunhofer IOSB's OPC UA modeller

Test results were quite clear. UaModeler is the most advanced tool for testing different model specifications, so it was used for main modelling tests. The other two tools have some features that UaModeler does not have but which are not needed in these tests.

4 PROCESS INTRODUCTION

This chapter briefly introduces two different styles of plants which will be modelled partly in the chapter 5. Both are one part of a copper refinery. First plant is a concentrator where copper is removed from other minerals using mills, hydrocyclones, froth flotation and filters. In this plant raw material is moved by conveyors, pumps and pipes. Concentrator is a continuous process.

Other plant is a tankhouse. Here copper is already refined to copper plates where copper purity is about 99%. This process uses electrolysis to make more pure copper about 99.9%. This plant's main modules are production cells, cranes, switches, preparation unit and stripping machines. All materials are transported by cranes in this plant. Cranes move anodes and cathodes into cells.

These plant introductions are done quite simply and shortly. There is no need to go any deeper into these techniques in this thesis. This is enough to understand what is modelled in chapter 5.

4.1 Concentrator

Concentrator is usually separated into three parts which are comminution, concentration and dewatering. First part, comminution, consists of methods like crushing, screening, grinding and classification. Second part consists of several froth flotation cell lines, few hydrocyclones and mills. Third part contains machines like thickener(s) and filters. All these parts are presented in Figure 4-1.

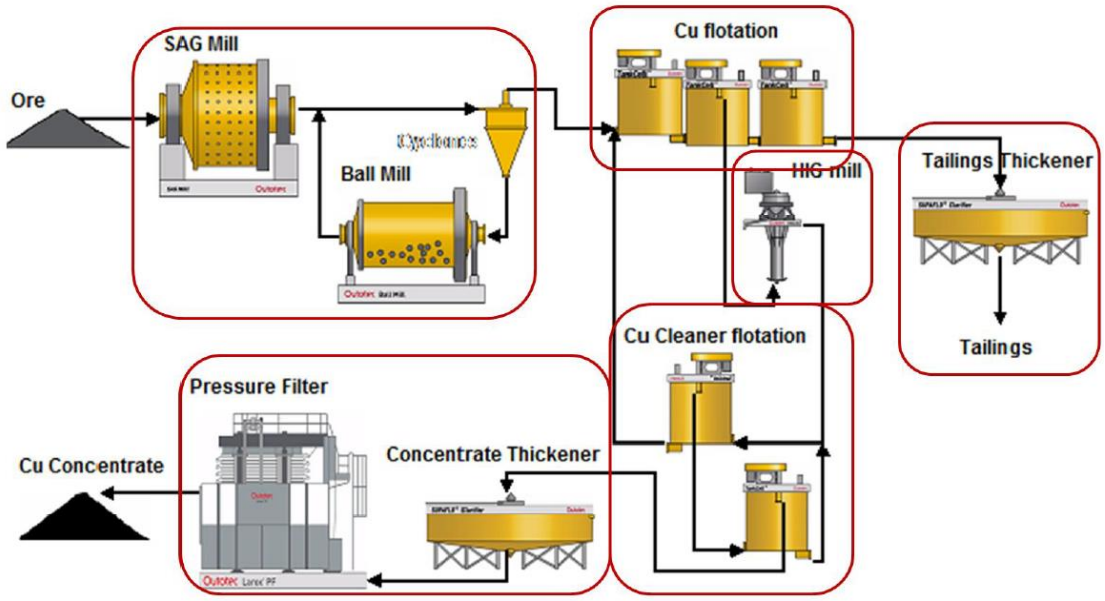


Figure 4-1: Concentrator

Comminution part of the process includes several different kinds of mills, tanks, screens and hydrocyclones. This section reduces ore size to small enough for the next part which is concentrator. Mills are usually cylinder mills which might be ball mills, autogenous mills or semi-autogenous ball mills. Ball mills use steel balls inside the mill to crush the ore. Autogenous mills use the ore itself to crush other pieces of the ore. Semi-autogenous mills use both of these techniques. There are also few options for balls, like hard pebbles and steel rods. One big difference between different mills is the mill cylinder diameter-length ratio. Mills are usually quite big, over ten meters long. Hydrocyclones are used for separating fine particles from the fluid and the coarse material using centrifugal forces. Overflow is passed to the next phase of concentrator and underflow is recycled back to the mill.

In the second part of the concentrator there are several different styles how to separate wanted minerals from other ore materials. These are for example froth flotation, magnetic separation, gravity separation, sorting and dense-medium separation. The most important minerals recovery process is the froth flotation. Froth flotation separates copper from other ore materials using certain chemicals like xanthate and air bubbles. Using these two together, wanted mineral is attached on the bubble's surface and floated over cells, when other minerals will be left on bottom of the flotation cells. These left overs are finally pumped out from the process and are called tailings. In Figure 4-1 there is also a thickener for tailings that removes water from solid materials.

Final part of the concentrator, dewatering, uses at first thickener to remove water from solid materials. Thickener uses gravity to separate water and other liquids and minerals to layers as overflow and underflow. After that minerals are transported to filters which use filtration where only fluids can pass through and also oversize solids are retained.

There are several analyzers in the plant to ease automation. These analyzers analyze for example particle size and mineral content with x-rays or with lasers.

In the concentrator process key values are overall recovery, grade of the feed, grade of the concentrate and new feed of the concentrator. Secondary values are mills' energy consumptions and other energy efficiency related values. Also reagent consumption is one measurement we are interested in. [Outotec self-study]

In concentrator plants many Manufacturing Operations Management (MOM) systems are often used. Usually these are made by end users or some other provider. Currently at least Manufacturing Execution System (MES), Laboratory Information Management System (LIMS) and Warehouse Management System (WMS) are used. The information model which will be done in this thesis will help to build maintenance systems like Asset Management (AM) and Computerized Maintenance Management Systems (CMMS). It can also be used with older MOM systems for example with LIMS.

4.2 Tankhouse

Tankhouse is a batch process. There are two types of tankhouses. First one is an electrorefining process (ER) and second one is an electrowinning process (EW). In first one all minerals are transported into the system via anodes. In the second one minerals are transported by electrolyte solution. ER tankhouse is presented in Figure 4-2. EW tankhouse differs slightly from the ER tankhouse. In EW tankhouse there is no anode scrap washer and instead of an anode preparation machine there is an electrolyte solution preparation. In both process anode and cathode plates are placed into process cells each after each. In both processes, process itself is done using electric current between anodes and cathodes. One process cell contains about 50 to 70 plates. Tankhouses can be used for refining several different non-ferrous metals like copper, zinc and nickel.

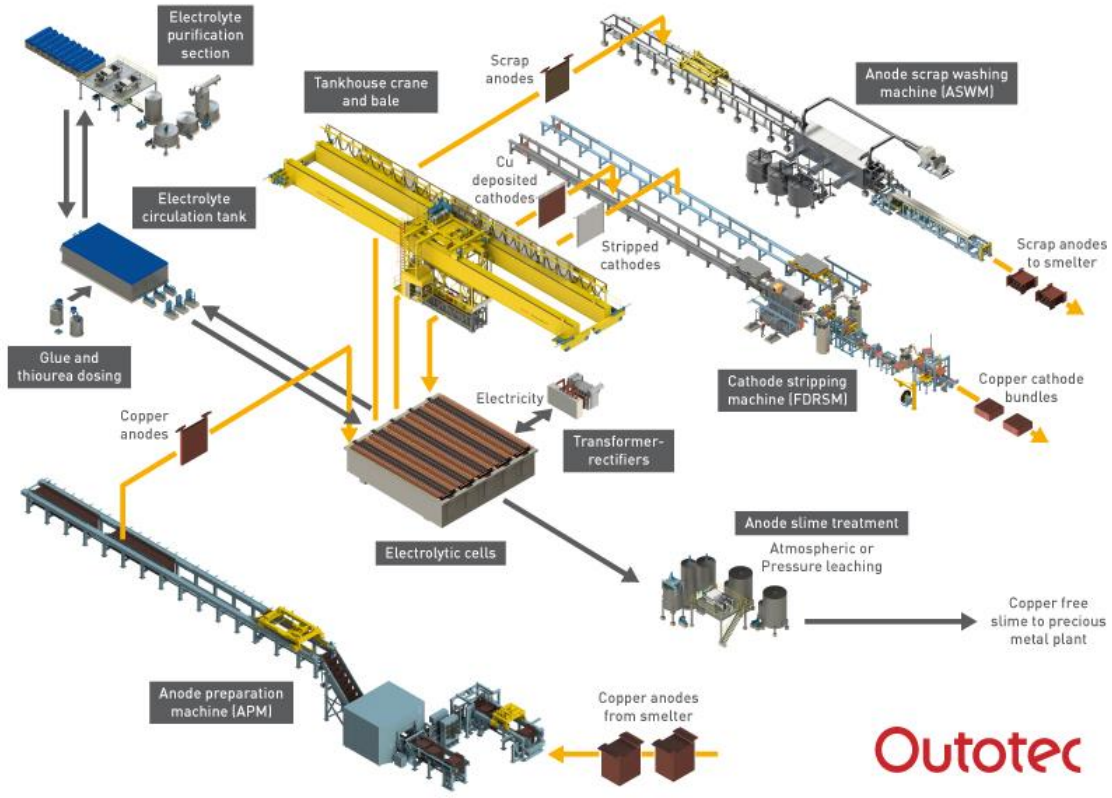


Figure 4-2: Example picture of ER-tankhouse's material flow [Outotec]

In ER tankhouse copper is brought to the process by anodes. Copper purity in anodes is about 99%. Copper from the anodes is moved to the surface of cathodes via liquid reagent. When cathode's surface has enough copper it will be replaced with another cathode. One Anode will last about 4-5 cathodes until all copper from it is removed. Full cathode will be transported to the stripping machine via cranes. In the stripping machine copper will be scrapped from the cathode plate. The cathode plate can be used again after this. Process cells in ER process are grouped into sections which each contains about 15-30 cells. In one tankhouse area there are about 10 to 30 sections.

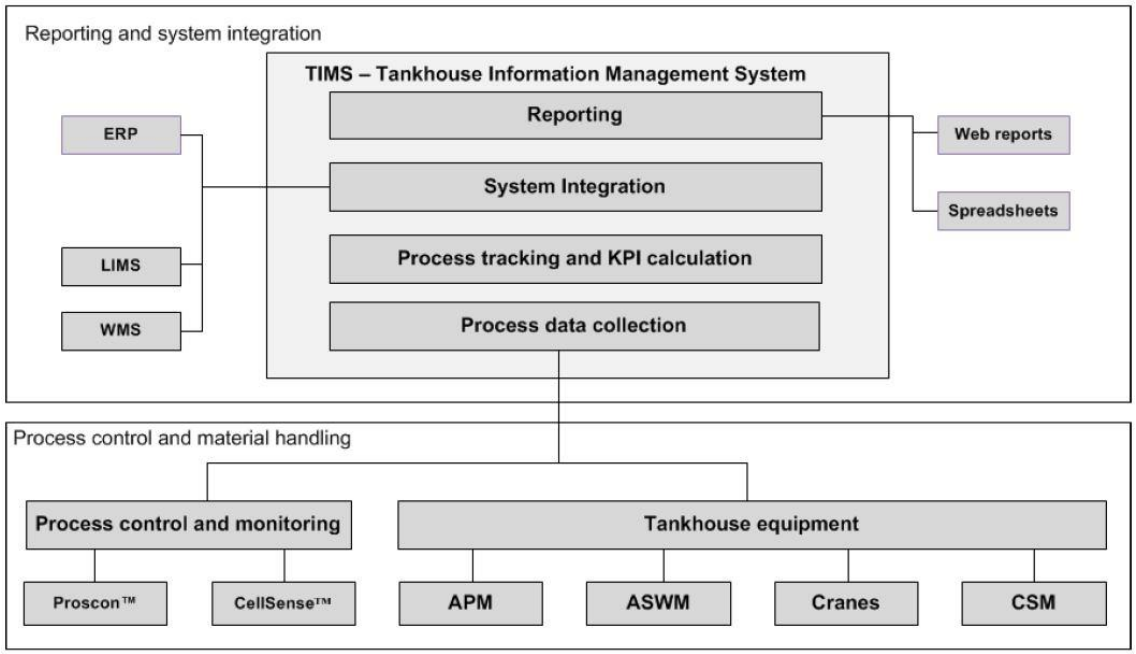
In EW tankhouse both anodes and cathodes are permanents. Anodes stay in the electrolyte almost always. They are moved only during maintenance. Refined copper is in the electrolyte solution from where it is extracted to the surface of the cathodes. When the cathode is full it is moved to the stripping machine where copper is removed from cathode as in the ER tankhouse. In EW process there is no need of grouping because each cell will be controlled separately.

The most important machines in both process types are cranes. Using these cranes efficiently means a lot. Cranes can hold all plates of one cell at a time and usually all cathodes are moved at the same time from one cell. [Kössilä 12, pp.21-25]

EW tankhouse key performance values are calculated per cathode cycle using values like electric current and time efficiency, production, additives consumption, quality of

the produced copper and energy consumption. These values are collected through several different systems. Tankhouse safety can be improved by tracking cell voltage and temperature. Cell temperature can also be used to optimize quality of the copper. [Rantala et. al. 10, p.5]

In tankhouses there are usually MES, LIMS and WMS systems implemented. Kössilä designed scheduling tool for ER tankhouse in his thesis work named Integrity. Integrity is based on General Electric Intelligent Platform Proficy product. One part of Proficy is based on ISA-95. The information model which is designed later in this thesis could be a tool for Integrity or parallel tool to help some other MOM level tools. Integrity can also called Tankhouse Information Management System (TIMS). One example of TIMS architecture is presented in Figure 4-3. [Kössilä 12, pp.15-16]



MES = Manufacturing Execution System, ERP = Enterprise Resource Planning, LIMS = Labratory Information Management System, ERP = Enterprise Resource Planning, WMS = Warehouse Management System, APM = Anode Preparation Machine, ASWM = Anode Scrap Washing Machine, CSM = Cathode Stripping Machine,

Figure 4-3: Example TIMS architecture for ER process [Larinkari, p. 5]

TIMS can be considered as a MES and it is used by ERP, LIMS and WMS systems. So far TIMS is only used with ER processes. TIMS is used for example for calculating several Key Performance Indicator (KPI) values. KPI values are for example current efficiency, tankhouse time efficiency, anode scrap percentage, energy efficiency and equipment availability. These example values help to figure out what kind of properties modeled equipment should have so that these KPI values can be calculated. [Larinkari, p. 6]

5 IMPLEMENTATION

This chapter introduces how models and hierarchies are modelled in practice. At first there is some general information about the modelling and then how modelling can be done with UaModeler. Example equipment from the concentrator and from the tank-house will be modelled next using UaModeler. UaModeler was selected for this task because it was more advanced than other tools. Both plants are modelled using ISA-95 levels and Equipment that has been modelled first. The last part of the modelling is to present how ADI model can be utilized with the Outotec Particle size analyzer. After modelling plants it is time to investigate how the OPC UA server can be used in the concentrator plant with other information systems and how data can be brought into the OPC UA server.

5.1 Modelling

This chapter is all about modeling equipment and plants. At first modeling capabilities are introduced explaining what OPC UA and UaModeler are capable of. After that two example plants and their typical Equipment are modeled using the ISA-95 styled structure.

At first there seemed to be no readymade ISA-95 model available in an XML format. That is why modelling needed to be started by creating an ISA-95 model. After some more searching an ISA-95 XML based model specification was found from the web site of the OPC Foundation. The first version of that specification contained some flaws but soon they released a better one. Modelling ISA-95 by oneself is quite hard and almost impossible. Some ISA-95 features need handling done by code behind the OPC UA server. For example decimal data type must be handled by application because OPC UA does not support decimal data type. [OPC UA ISA-95 13, p.31] This is the case also with the official version of the ISA-95 model specification. It may be concluded that all ISA-95 features are not supported by the OPC UA.

All equipment are inherited from the ISA-95 *Equipment Type*. First there is inherited *Outotec Equipment type* which is a base object for all Outotec's Equipment. After that there is an inherited Unit Type which corresponds to an ISA-95 Unit Level. An example of the type hierarchy is presented in Figure 5-1. Every Unit level Equipment are inherited from the Unit type object. Now there is a possibility to make other general Equipment by inheriting from the Outotec base Equipment Type. These can be for example

higher level Equipment like flotation or concentrator. First step to model ISA-95 styled plants is to make these new Equipment Types.

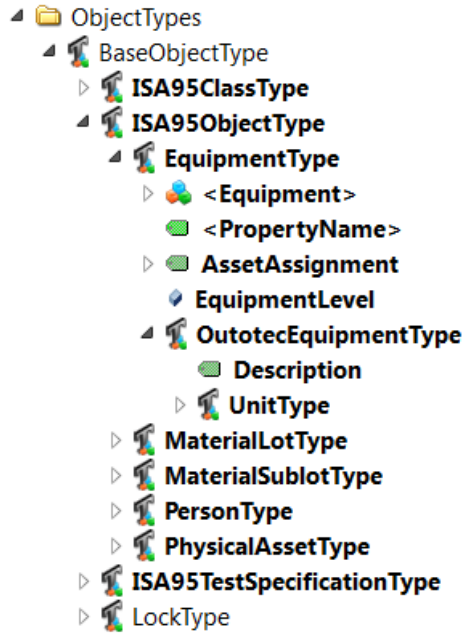


Figure 5-1: General Outotec type hierarchy

Chapter 3 introduced where to find information for modelling these new Types. Now there is a problem to distinguish which of the Equipment are Unit level equipment and which are upper level equipment. It is first good to understand the hierarchy of the plant and meanings of the ISA-95 levels. Level descriptions can be read from the part 1 of the ISA-95 standard [ISA-95.01 10, pp.25-30]. Level selection can be done by starting from the top of the ISA-95 level which is Enterprise and lowering down level by level and finally finding out which are correct levels of all equipment. For the example plants in this thesis Enterprise is the same for both. It is *Outotec O&M*. After the Enterprise level there is a Site level. Site level equipment are named based on the plant location. Both of the example plants are modelled separately in their own chapters later on. Levels below the Site level are also gone through in those chapters. Work center and work unit levels are hardest to figure out. Other levels are usually quite clear. Work centers are typically scheduled by the level 4 or level 3 functions, whereas Work Unit equipment are scheduled by level 3 functions. This may help to figure out the levels. Work centers also have well-defined capabilities and capacities which are used for level 3 functions.

One notable feature of the OPC UA is ‘is Historizing’ property which can be set for all variables. This doesn’t mean that OPC UA itself has some kind of historian. It just tells that this variable has some history data. OPC UA clients can call a method which returns history data for these variables. Integrator must implement this handling into the UA server and use some database where history data is located. This feature must be considered when modelling Types if some Equipment always has some variable containing history data.

When modelling these instance hierarchies it should be taken into account that in ISA-95 Equipment model there is a *Made up of Equipment* relation between two Equipment. Corresponding reference in ISA-95 model specification can't be used in UaModeler. UaModeler only offers *Has Component* or *Organize*. This may also be a flaw in the OPC UA ISA-95 model because if *Made up of Equipment* would be inherited from *Has Component* or *Organizes* instead of *Aggregates* it should be an option in UaModeler.

Different kinds of plants consist of quite a different range of equipment. Because of this manufacturer specific type namespaces should be one for each type of plant. For example own type namespaces for the concentrator and the tankhouse. This way it is easier to manage different kind of plants.

Next is presented how to model new object types. A good foundation for the new types where they should be inherited from must be chosen. For Equipment this is an easy task. Now, when a type is chosen new types must be named. Type naming is otherwise free but it cannot begin with any other than a letter and must end with the type. Next, properties will be added into the type. When adding a new property reference type must be changed to a correct one. In this case it must be *HasISA95Property*. Property's variable type can be anything that is inherited from the *Equipment Property Type*. In this thesis OPC UA *Data Item* Types has been used instead of *Equipment Property Type* in specific situations. *Data Item Type* structure is very good for any data which have for example engineering unit (kW, °C…) or ranges. For analog signals *Analog Item Type* is a good choice and for discrete signals *Discrete Item Type* is a good choice. They can be again modelled inheriting *Equipment property type* but there is no point for that. Final step is to choose which namespace this new object should be added to.

Same rules can be used for Physical assets. Physical asset has few properties already, like *Physical location* and *vendor id*. Almost always there are some model and serial number variables in Physical asset. Designer can add these to object which is inherited from the Physical asset type for example to Outotec physical asset type. All Outotec specific Physical asset types are inherited from this specific type and they all have these two variables.

Code maintenance is quite a challenge. Here are some things which should be considered when making an UA server. If Type needs tailored code inside the generated code there might become problem when Type is updated. The integrator should inherit all Types in the code and make modifications into these subtypes. This will prevent most of the problems. This only works in certain situations but if Type structure will be modified radically even this will not help. Maintaining Types should be done with good care and the structure of the Types should be done by thinking of maintenance. Well build Type inheritance will help with this problem. Also specifying how Types should be

used eases this task. For example if all clients are made with using agile code this helps them not to get broken when Types are changed. To be able to create agile code structure of the Type must come with the measured data. These are just few examples how this can be handled. Code maintenance is a much bigger issue than these two examples and should be done carefully.

5.1.1 Modelling example plants with UaModeler

After a new project is created into the UaModeler a designer must import ISA-95 model specification using an XML file. This will transform the file into UaModeler's file format. If some item is added to the ISA-95 namespace it will affect only to the transformed file. Only regenerating the XML file will affect to the original file. Regeneration can be done using the XML export function if needed. Next a designer can start to inherit new Types under the Types folder. The *ISA95 Object Type* is located under the *Base Object Type*. Others can be found by exploring namespace or checking them from the specification. ISA-95 Type names are bolded in the UaModeler so that they can be found more easily. Designer's own Types are also bolded.

When starting to make new instances under the Objects folder designer should make some folders for different ISA-95 resources. This is just for clarity. Example folders can be *Equipment set, Material set, Person set* and *Physical asset set*.

5.1.1.1 Modelling Concentrator

The information model for concentrator is done to create more structured information. Equipment types standardize the structure of the equipment. Each plant that is done using these equipment models is using the same structure. This makes it easier to compare the data of different plants. This also standardizes the way how things should be done. Equipment names and equipment levels of the concentrator are listed in Table 5-1. Their typical variables are presented after the table.

Table 5-1: Equipment of the Concentrator and their ISA-95 levels

Name	**ISA-95 level**
Location name of the site	Site
Concentrator plant	Area
Mine area	Area
Utilities	Area
Grinding	Production Unit
Flotation	Production Unit
Dewatering	Production Unit
Filter	Unit
Flotation cell	Unit
Flotation circuit	Unit
Froth sense camera	Unit

Hydrocyclone	Unit
Mill	Unit
PSI analyzer	Unit
PSI sampling line	Unit
Pump	Unit
Tank	Unit
Thickener	Unit

The site level of the concentrator can be named for example by the location name of the site. Next levels are Area, Production Unit and Unit. In concentrator Areas can be for example *Concentrator plant, Mine Area* and *Utilities*. *Mine area* and *utilities* are left for less attention and focus will be on the *Concentrator plant*. Key values for the concentrator are *New feed, Total energy* and *Raw water*. In the *Concentrator plant* Production Units are *Grinding, Flotation* and *Dewatering.* Grinding key values are for example *energy, Kw/Ton* and *Throughput* values. Flotation values are for example *Concentrate grade* and *Recovery*. Dewatering key value is *Concentrate out flow*. These kinds of values can be used by different kinds of ISA-95 activities. This hierarchy is presented at the left side of the Figure 5-2. Unit level equipment has gotten a lot of attention because every concentrator consists mostly of these modelled equipment. They can be used more easily again because their structure and values are the same for every plant. This also helps comparison between different plants. Unit level equipment are often part of some control circuit. These circuits are for example *Primary Grinding* and *Secondary Grinding* which are not Production Unit level equipment and not Unit level equipment. They are rather something between these levels. These circuits consist of several Unit level equipment. If some Unit level equipment must be grouped by circuit it can be done using some property like *Circuit Name*. This property should be optional in the *Unit Type* and value should be the same for all the equipment in the same circuit. The hierarchy of the types is shown at the middle of the Figure 5-2.

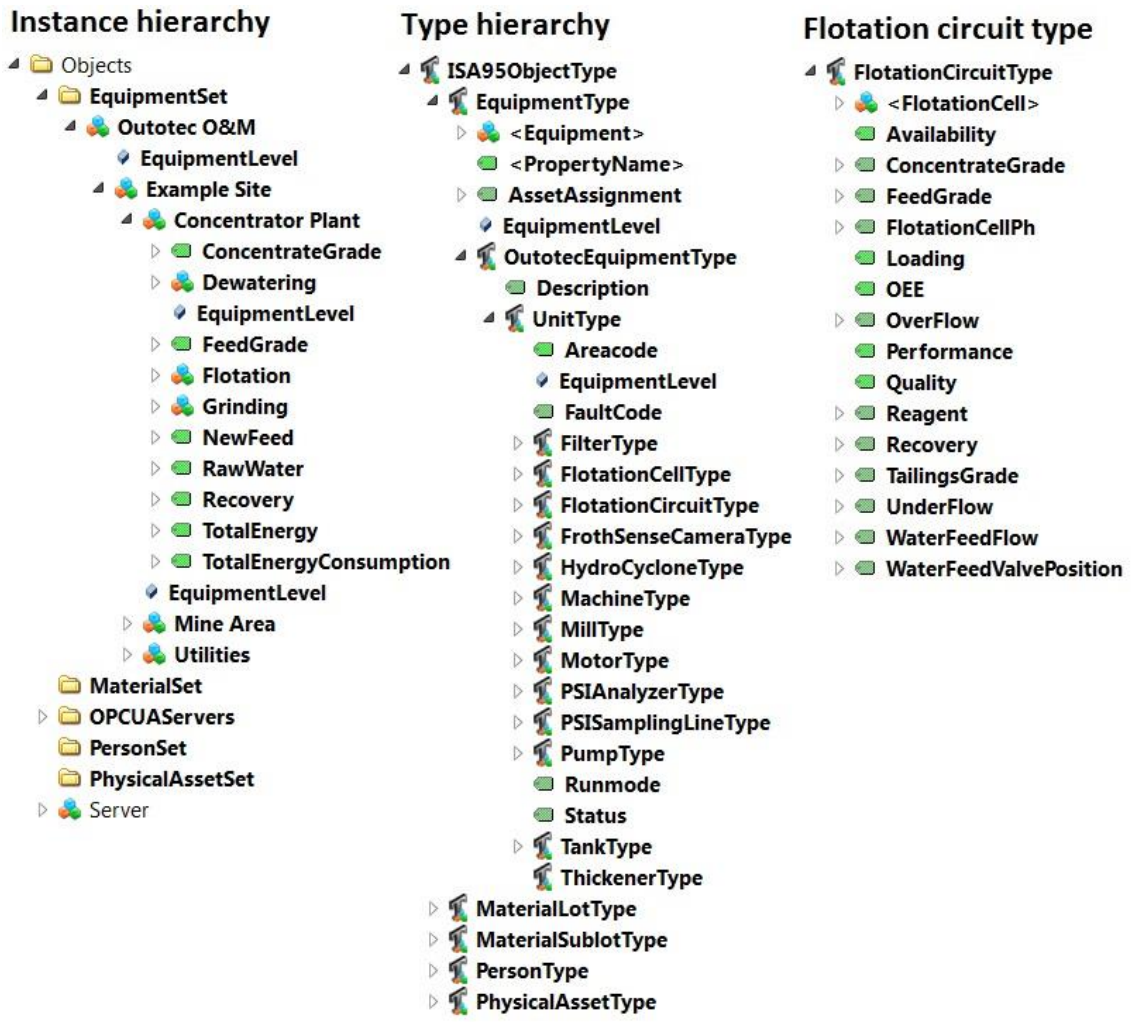


Figure 5-2: Concentrator example hierarchy, types and example equipment type

At the right side of the Figure 5-2, there is an example of the *Flotation circuit* equipment. *Flotation circuit*s always consist of at least two *Flotation cells*. In the Figure 5-2, on both sides of the "FlotationCell" text there are < > brackets meaning that its modelling rule is placeholder. Placeholder modelling rule makes adding of *Flotation cells* into the *Flotation circuit* easier. The UaModeler has a good feature for modelling rules. There are two buttons, one for normal modelling rules and one for placeholders. Components that have a normal modelling rule can be selected from the optional components dialog. This dialog is presented in the Figure 5-3. In this dialog variables and objects can be selected from different levels of the type hierarchy. Dialog also shows mandatory components which can't be deselected. Components that have a placeholder modelling rule can be added through the placeholder components dialog. The placeholder components dialog consists of one list for each placeholder component. New placeholder components can be added just by adding new row to the corresponding list and give it a name. The Figure 5-3 also shows how additional information can be added to the *FlotationCellPh* variable whose type is analog item. At the right side of the figure there is shown that in addition to the list for *Flotation cell* there are also lists for *Equipment property type* and *Equipment type*. These come from the *Equipment type*.

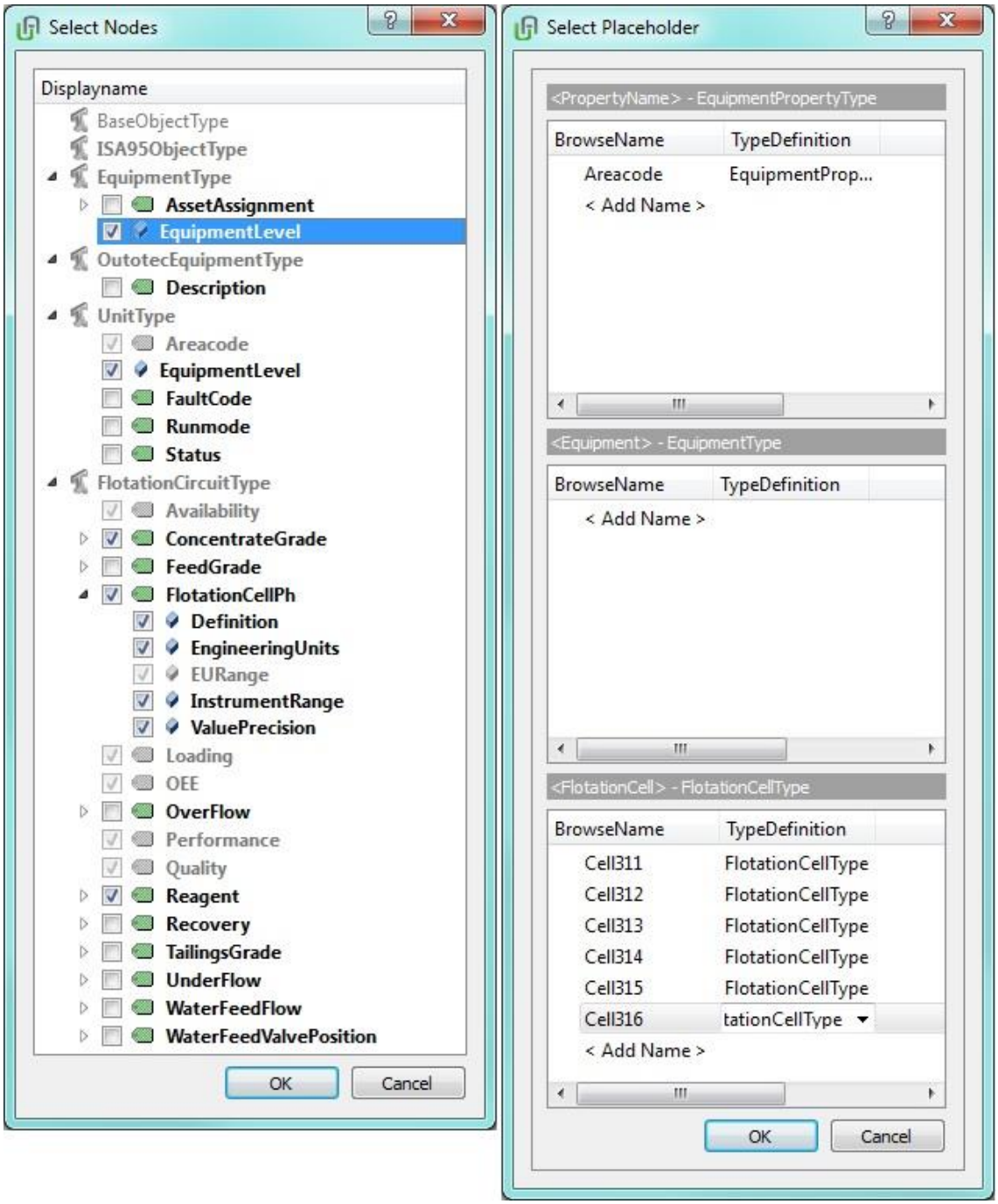


Figure 5-3: Optional component selection for flotation circuit type

Flotation, Grinding and dewatering types can use same kind of design that was used with *Flotation circuit* and its *FlotationCell*s. Flotation, Grinding and dewatering types consist of several components which all have the placeholder modelling rule. These components could be for example unit level equipment types like *MillType* and *Hydro-cycloneType* mentioned earlier. This makes adding equipment faster and reduces human errors when offered Types are pre-selected.

Until now there have only been new designed equipment types and their variables. Some new reference types can also be added which can describe how the plant is work-ing. For example *FlowsTo* reference type can be added. Its inverse name should be

FlowsFrom. It can be used to describe the direction in which the slurry is moving between two Equipment. With this information user interfaces can be done for the slurry flow. Other scenario is to optimize how long each step takes and is some machine faster and efficient than another. This kind of use needs a property which tells how long it takes for the slurry to go through the equipment. This is presented in the Figure 5-4. In this figure there is a *Pass through time* variable for this use. *Flotation cell*s are part of the *Flotation circuit* and thereby part of the instance hierarchy.

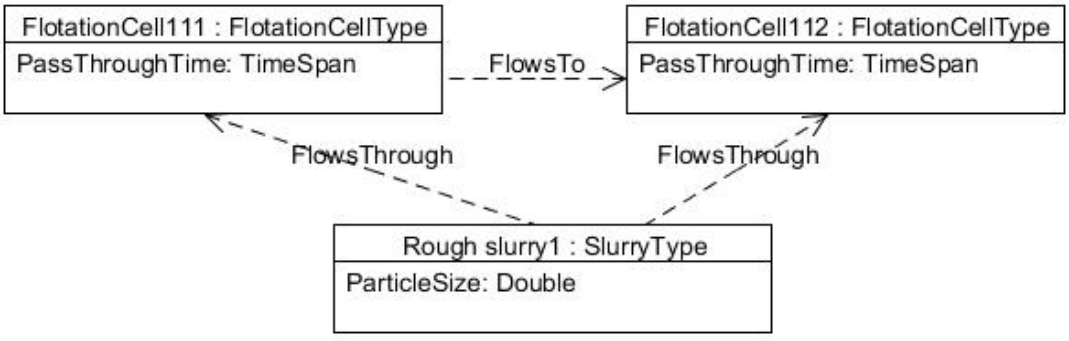


Figure 5-4: Example use of own reference types

Another reference type could be added between the material and the equipment. This reference name is *Flows through*. This reference tells what material is going through the particular equipment. This is presented also in the Figure 5-4. *Slurry type* is inherited from an ISA-95 material type. It has a *Particle size* variable that tells how big particles are. This information can be updated with the help of the particle size analyzer. In this figure same material is going through both equipment because there is no particle size analyzer between cells and the slurry does not change that much between cells. *Rough slurry1* is a part of the *MaterialSet* in the instance hierarchy of materials.

5.1.1.2 Modelling Tankhouse

The ISA-95 hierarchy model for tankhouse has been done earlier in many works [Kössilä, p.38] [Rantala et. al. 10, p.4]. This thesis is using these works for basis for the hierarchy model. Equipment names and equipment levels of the tankhouse are listed in the Table 5-2. The tankhouse instance hierarchy is presented at the right side of the Figure 5-5.

Table 5-2: Equipment of the Tankhouse and their ISA-95 level

Name	**ISA-95 level**
Site location name	Site
Tankhouse	Area
Utilities	Area
Mine Area	Area
Cell Area	Process cell
Crane system	Process cell
Rectifiers	Process cell
Anode Preparation Module (APM)	Process cell
Anode Scrap Washer (ASW)	Process cell
Cathode Stripping Machine (CSM)	Process cell
Cell group	Unit
Rectifier	Unit
Crane	Unit
Switch	Unit
Cell	Unit

The tankhouse site level can be for example site location name. The area level equipment are *Tankhouse, Utilities* and *Mine area*. The last two areas are the same for the concentrator also. Utilities include for example equipment for preparing electrolyte for an EW tankhouse. We are not interested in these in this thesis. Main focus is on the *Tankhouse*. An example tankhouse is an ER tankhouse. Lover levels are Process cell and Unit because tankhouse is a batch process. Batch process has its own expansion for ISA-95 levels and adds two more levels. This expansion is called ISA-88 and it brings *Equipment module* and *Control module* levels. These levels can contain themselves. This way any plant can be modelled in the desired depth. These levels can be used for *Cell* and *Switch* equipment but in this example they are Unit level equipment.

In the Figure 5-5 there are only 4 sections in the *Cell Area* to save space. In a real process there are usually over 20 sections.

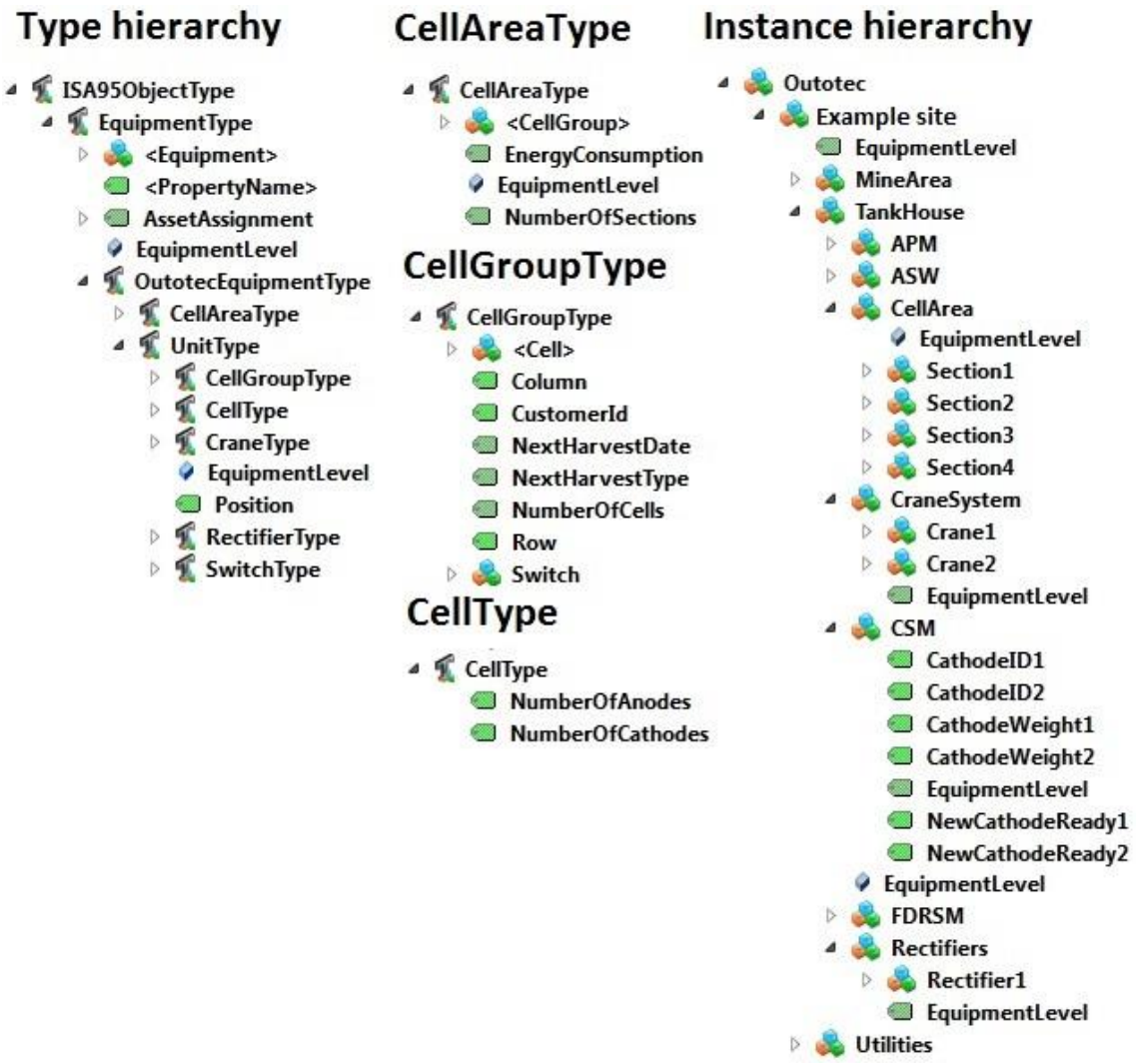


Figure 5-5: Example types and hierarchies of the tankhouse

Figure 5-5 also shows what kinds of types are modelled for the tankhouse. One notable difference compared to the concentrator is that cell area type is modelled which is the process cell level equipment. In the concentrator there wasn't any other equipment modelled than just the unit level.

In the Figure 5-5 there are also example equipment models about different levels of the cell area. The same modelling rule technique is used than in the *Flotation circuit* in the concentrator. *Cell area* has a *Cell group* object with optional placeholder modelling rule. *Cell group* has the same modelling rule for the *Cell. Cell group* also has one *Switch*.

In the tankhouse there is often a spare rectifier. This can be modelled using the Equipment Physical asset relation. In the Equipment hierarchy there is always just one rectifier for one cell area but in the physical asset hierarchy there are two rectifiers of which one is in use and the other is not. This can be modelled using the *Implemented by* reference between the rectifier equipment and the physical asset which is in use. The same kind of design can be utilized also elsewhere if there are some spare parts which may be

replaced often. Sometimes tankhouses expand with another cell area. Then they start to use a spare rectifier but still this kind of design can be used. The new Cell area just needs its own *Rectifier* equipment and its *Implemented by* reference starts to point to the spare rectifier physical asset.

Another place where this *Implemented by* relation can be used is with the cathode and anode tracking. Here cells, where cathodes and anodes are located in the process, are Equipment that includes other lower level Equipment. These lower equipment are slots for every cathode and anode inside the cell. Now cathodes and anodes are physical assets that implement these slots. Now tracking can be done with the asset assignment that equipment and physical assets have. Cathodes and anodes are grouped just before they are moved to the cells. These groups can also be used with the tracking when the destination cell is pointing to this group with the *Implemented by* reference. Tracking might be impossible because one tankhouse has several tens of thousands of cathodes and anodes. There are also several tens of thousands of locations where cathodes and anodes can be in the cells. If this is needed to be implemented this way for tracking purposes OPC UA should be tested for efficiency so that OPC UA can hold this much information and change it quite often without any lack of performance.

One new reference can be used with the tankhouse. Its name is *Is Powering* reference. It can be used with rectifiers and cell areas. This reference means that the rectifier is powering some cell area. This can be used if there are several cell areas in one tankhouse.

5.1.2 Using ADI model for modelling Particle Size Analyzer device

Outotec Particle Size Analyzer (PSI) devices are used in the concentrator for controlling the plant more efficiently. The particle size is important in many phases in the concentrator. For example correct sized particles are easier to get attached to the air bubble surface in the flotation. Particle size information is used here to control the earlier phase that is grinding. This also helps the plant to get a maximum throughput when mills don’t need to crush the ore into too tiny particles. Other sampling points are right after the flotation, one for tailings and one for the slurry. These also help to control the grinding if tailings have too big particles that are not attached to the bubble surface.

The PSI devices use a couple of different mechanisms to detect the particle size. One is to physically press the slurry against a wall using a linear actuator and a sensor that detects how deep the actuator goes. The calibration is simple; just make one movement with an empty analyzer and measure how deep it can go. Other mechanism is to use a laser diffraction. The laser diffraction is based on the intensity distribution measurement of a coherent laser light scattered by the particles. It uses Mie theory for calculating particle size from the scattered light. [Outotec s2, p.2] The Mie theory uses optical properties of the sample and the dispersant to calculate sample diameters. [Wikipedia]

In the ADI companion specification's annex A [OPC UA ISA-95, p. 102] there is an example of a particle size analyzer which uses the laser diffraction technology. In this thesis this example was chosen to test how it can be used with the PSI device.

For PSIChannel configuration there is a property for a *detector count*. In the PSI device there is only one detector. In the channel status there are four parameters which are presented in the Figure 5-6. In the stream Acquisition Settings there are several parameters which are also presented in the Figure 5-6.

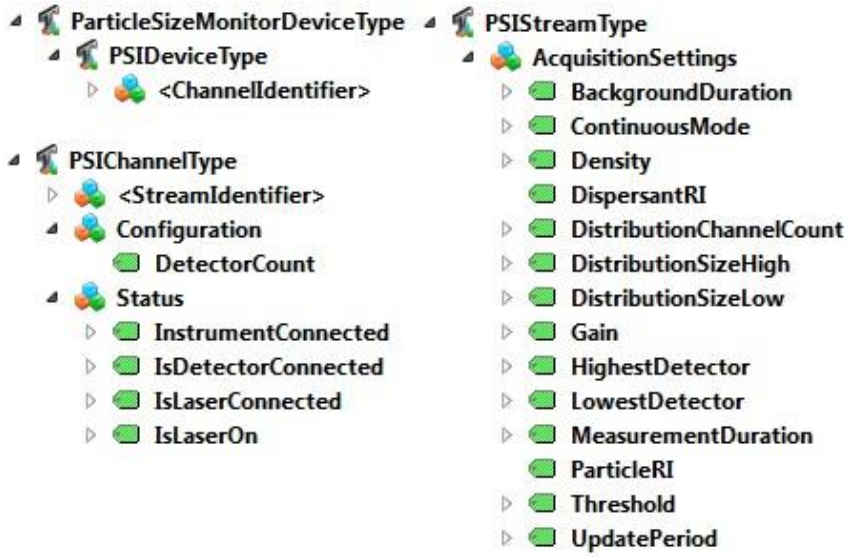


Figure 5-6: Example type of Outotec PSI- device

Figure 5-6 shows that the *PSIDeviceType* is inherited from the *ParticleSizeMonitorDeviceType*. It also shows that *ParticleSizeMonitorDeviceType's Channel type* is overridden by a new made *PSIChannel type. StreamType* is overridden by a *PSIStreamType*. This way they can be used correctly when making a new object.

Figure 5-7 presents the structure of an example PSI device. It has one channel which have three streams. These streams are presented in the Figure 5-8. Analyzer itself has accessories to control which stream should be analyzed. Channel has accessories to control the laser and the detector.

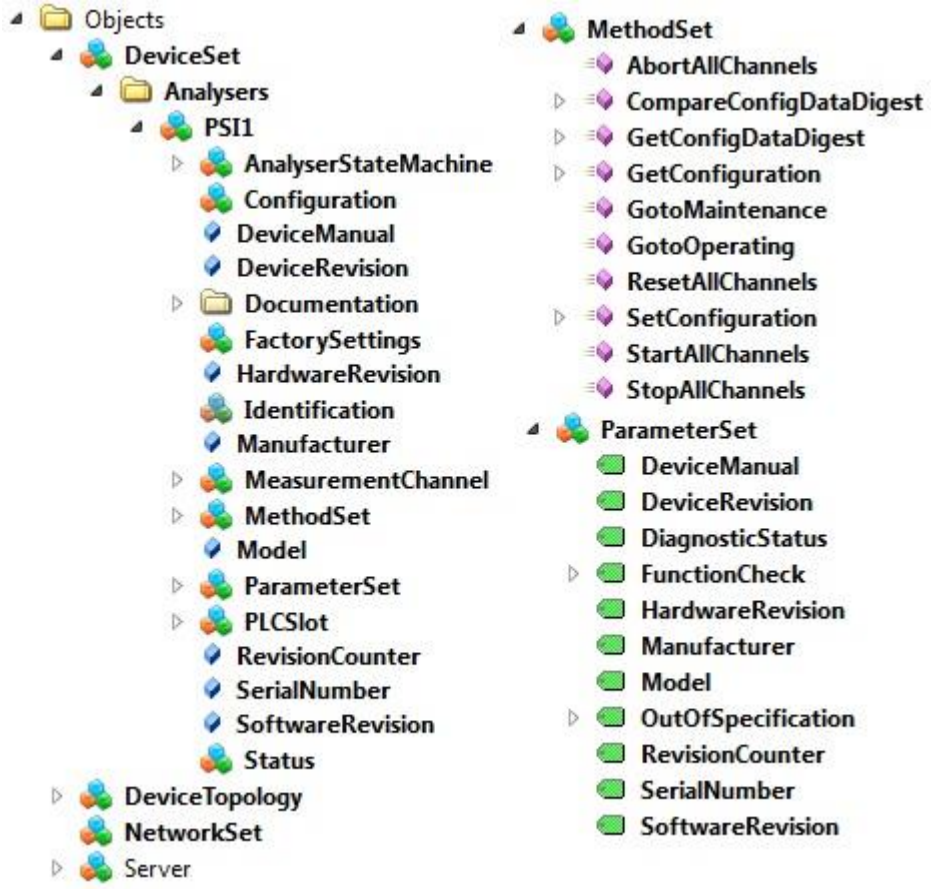


Figure 5-7: Example picture of PSI device hierarchy

Right side of the Figure 5-7 there are example methods and parameters which are mandatory for every Analyzer device. These are inherited from readymade types from DI and ADI companion specifications.

Figure 5-8 shows what kind of properties and methods channel and stream have by default. These are also properties from DI and ADI specifications.

Figure 5-8: Example information about one channel and stream

The PSI device has now been modelled successfully. Next step is to connect it to the devices. The PSI user interface is running on a Linux based operating system. The user interface itself is done using Java Script. There are two possible ways to make own UA

server for the PSI. The first option is to use new Java Script version of the OPC UA which is not officially tested by the OPC foundation [npm]. With this option there might be some compatibility problems if this version is not done properly. Java Script version of OPC UA is under development and development is done by the community. Another version of the OPC UA Java Script has been done by the Unified Automation but it is not released yet [OPC&MES day]. The second option is to make an UA server using the UaModeler and generating a C++ version out of it. This version can be run on Linux. This option also needs a way to connect to the data. For this problem some web service interface can be used. This interface should use some general technique like the Web Service Description Language (WSDL).

After the PSI device starts to provide an OPC UA interface from it all parameters can be set and control can be done remotely if needed. Some features should be set only locally and not provide all functionalities for the remote users. One useful example for this is with mathematic models parameters which can be updated through a remote interface after laboratory tests. This way operator does not need to go to every PSI device locally but set them through some remote interface. The OPC UA server in the PSI device can also be thought as Outotec first step towards Internet of Things.

5.2 OPC UA architecture in the Concentrator plant

After the plant information model is done it should be connected to other information systems. The concentrator plant usually has a historian database where history data is collected through the classic OPC. This way signals don't have any additional information than name, type and description. All signals are on the same level. This makes it hard to understand the structure of the data. The information model which was built in the earlier subchapter can add more information into signals and also add metadata between signals.

Figure 5-9 presents how the new OPC UA server can be located in the old plant's information system. Classic OPC Gateway is already in the old system. It is used for collecting data into the historian database. The first step when adding an OPC UA server into the information system would be to set an OPC UA wrapper on in the classic OPC server. This is the fastest way data can be imported into the OPC UA server. This can be done for example using Kepware's KepServer that has this wrapper feature. This was not tested in this thesis but enabling it can be done very easily. The endpoint must be enabled and required security should be set for the endpoint.

The second step when integrating the OPC UA server into the old system would be to start to use lower level systems which provide the UA interface. There are already several PLCs and other field devices which can do this. This way all OPC UA functionali-

ties can be utilized when all parties have OPC UA implemented inside them without any wrappers.

The third step can be done utilizing the Internet of Things idea with field devices that provide UA interfaces. These field devices are for example intelligent valves or sensors which can tell their ID and other information. This also makes the integration easier with the ISA-95 physical asset model when devices themselves know this kind of information that physical assets usually need.

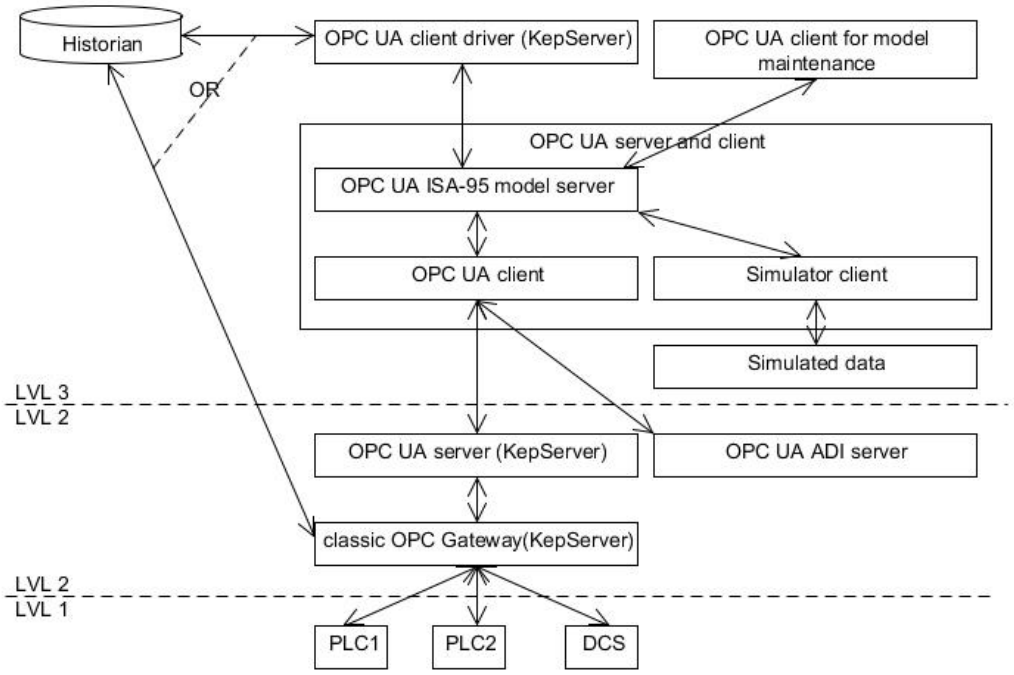


Figure 5-9: OPC UA architecture plan in Concentrator plant

In the Figure 5-9 one ADI server is added to clarify that the OPC UA server can be connected with several UA servers using OPC UA clients. This way all data is coming from other OPC UA servers or can be updated by upper level systems. Simulation is also added to the picture to present how simulator client can be used as an OPC UA client to get simulation data from the simulator.

Historian database can collect data through an OPC UA ISA-95 server. Historian data can also be shared through an OPC UA server. OPC UA also enables that historian database can also be situated on a centralized data center located outside of the sites. The purpose for this historian database is to collect big data from different sites into one data center. This way data from different sites can easily be compared and analyzed from the big data. Equipment models also ease effort for comparing data between different plants because different sites will use the same structure and types. There are already some OPC UA historian servers in the market. For example Prosys's Historian which can collect data from several different OPC UA servers into one centralized historian database. That historian uses SQL database for saving the data. [Prosys]

For maintaining the OPC UA address space there should be one client for this job. There is already one tool designed for this use. It is called Easy95 which has been mentioned earlier. Updating the address space this way requires that server has functionality to save these changes into some file if server goes down. It is also possible to maintain the address space using the UaModeler but it is not meant for this. However, this can be utilized by exporting a new XML file from the UaModeler, shutting down the server, replacing the old XML file with the new one and starting the server again. Although, XML should be used only for small address spaces because loading the address space will become slow with huge address spaces. Big address spaces should be saved into the binary format.

The ISA-95 is designed for MES and ERP use and that's why it is needed to be integrated with these systems. Jouko Virta's research result in his thesis was that an OPC UA to XML converter must be used when integrating OPC UA and ISA-95. Same idea can be used here when integrating MES level systems with the OPC UA server. Because of the ISA-95 companion specification was used conversion might be a little bit easier. Converter transforms OPC UA objects into an XML file which follows the B2MML schema. [Virta10, p.44]

5.3 Connecting process data to OPC UA server

This chapter covers how process data can be connected with the OPC UA server. Several different hierarchies can be used for connecting data from one model to another. These hierarchies can be for example the ISA-95 hierarchy and other is for example the PLC hierarchy. Figure 5-10 presents this kind of design. In the Figure 5-10 *Agitation tank*'s level measurement is linked to the PLC hierarchy's *Channel1* object. PLC hierarchy consists of information how an OPC UA client can connect to another OPC UA server. In this example connection information is in *Address* and *Type* variables. The PLC hierarchy can also be located in another OPC UA server and then this connection information is not needed. There is a hierarchy of the PLC network under the *PLCServer* object. There can be several different PLC networks. Each network can consist of several distributed IO modules. These IO modules have different kinds of IO cards where IO signals are connected to. Every IO card may have several channels. The example channel *Channel1* consists of information where data can be fetched from using an UA client or it can already contain a value.

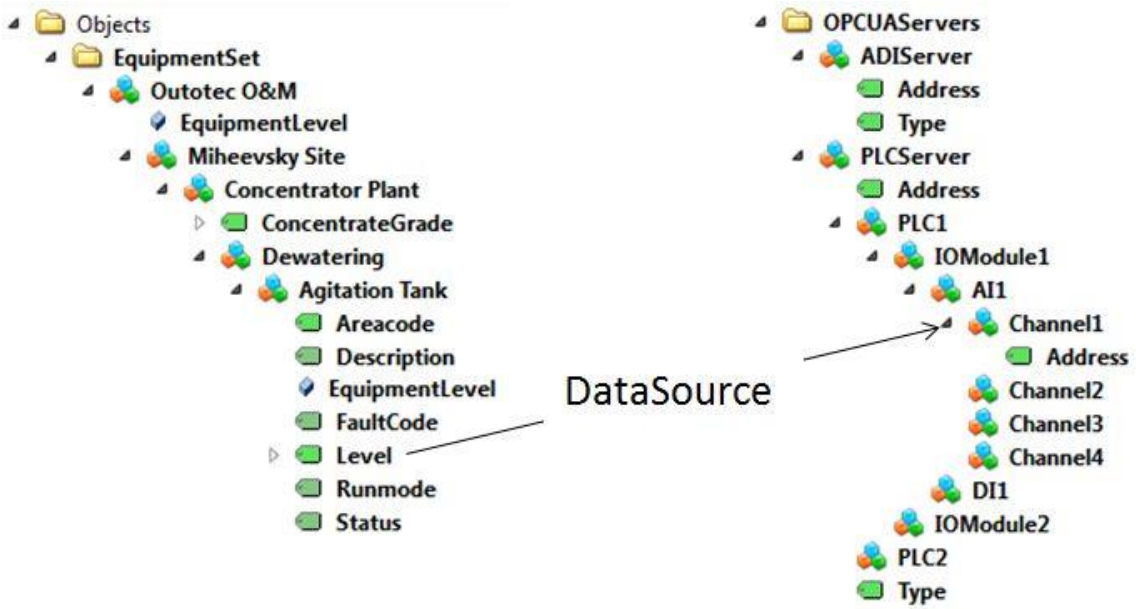


Figure 5-10: OPC UA data connection option

This linking can be done when modelling but an efficient way is to make a user interface for this. The user interface could have a feature where a designer can browse different UA servers and pick one. The UA client will be connected in real-time and the designer can now browse a namespace of a different server and select a node where data should be fetch from. This way the process data can be brought to the ISA-95 server by looping every *data source* reference and fetch data from another UA server using an UA client. This way OPC UA servers can be parallel to simulation and changing between the simulation data and the process data can be done easily. If simulation is needed there should be information in the same manner than in the *PLCServer* object. There should also be a *simulation source* reference and own hierarchy for this use.

If history data must be integrated into the UA server there should also be own hierarchy for the historian. That hierarchy includes information where to fetch history information of one variable. There must then be a new reference to present this. It could be a *History Source* reference.

6 RESULTS AND DISCUSSION

This chapter presents what the results are from the earlier chapters. Results are collected in the Table 6-1.

Table 6-1: Results of work

Goal	**Result**
Equipment for the concentrator	Several examples were done
Equipment for the tankhouse	Several examples were done
Hierarchy for the concentrator	Done with the required depth
Hierarchy for the tankhouse	Done with the required depth
Modeler tool test	UaModeler was found to be the most suitable
ADI model plan	ADI can be used with the PSI device
Integration plan	Integration can be done in several different ways
ISA-95 use	Should be used between MES and ERP interface
CAEX use	Should be used with SCADA

Example ISA-95 equipment types were modeled for the concentrator and for the tankhouse. These types were typical equipment that these plants usually contain. OPC UA modelling rules can be used effectively with these types, especially placeholder rules. These rules can speed up the next phase of modelling the plant. The next phase is to model the instance hierarchy using the modeled equipment. The UaModeler was the main tool when modelling was tested with these example plants and their equipment. Other possible tools were also tested and confirmed what they were capable of but they were not as good as the UaModeler for this kind of work. The results of the modeling were two Outotec specific type models for each plant and also two hierarchical instance models of these plants. These hierarchical models use levels from the ISA-95 and one part of this job was to identify different levels of the plants. Hierarchical models were not finished for all different areas. In the concentrator main focus was the concentrator area and in the tankhouse the tankhouse area. Main goal for these tests were to discover what this tool and the ISA-95 model were capable of. The result was that the selected tool was a little buggy but some of these flaws were already fixed during the thesis work. The OPC UA itself enables a very flexible way to model any equipment or plant so it was very useful for this purpose. The ISA-95 helps integration with the MES level systems. The whole modelling test can be thought to be a guide for modelling plants using the ISA-95 and the OPC UA.

Connection to the process was not tested and implemented in this thesis but some research was done how process can be connected into the ISA-95 model. The result was that using an OPC UA client this can be done with few different options. This requires that classic OPC or devices offer the OPC UA interface. Also the architecture of the plant was researched and what relation the OPC UA server has with other tools and software in the old plant. There was a problem how to maintain servers address space especially when server goes down. This was solved by suggesting implementation of an XML or a binary file generation into the OPC UA server. XML should be used only with smaller address spaces and binary format with bigger address spaces.

In the implementation part several different new reference types were introduced which can be used with the concentrator and the tankhouse. These were for example a reference for flow tracking of the concentrator and a reference for the rectifier powering certain cell area in the tankhouse.

The ADI model was also introduced and tested in this thesis. As a result was a plan how the ADI model can be used with the Outotec Particle size analyzer. The ADI model introduced several building blocks which are used when modelling an analyzer. These building blocks were used when an example model was made.

7 CONCLUSIONS

Plants are still mainly using the classic OPC to connect software into the devices. The classic OPC has many flaws which can be eliminated using the OPC UA. The OPC UA adds many features that were missing from the classic OPC but that were needed for a long time. These are type modelling, additional information for signals and good security features. There are also many other features beside these. Additional companion specification models which can be added above OPC UA information model can be used to standardize models between different vendors of the same field. This thesis studied the ISA-95, CAEX and ADI companion specifications, although the OPC UA model specification for CAEX was not ready when writing this thesis. The ISA-95 takes a leading part when testing modelling with the example production plant.

Before modeling could be done there was a part where the example plants were introduced briefly. The structure of these plants and what kind of equipment these plants contain were explained. The factory structure was checked through the PI diagrams and with engineers. There was a set of PI diagrams with different level of information. The structure of the plant and the different ISA-95 levels were not that easy to identify from the PI diagrams. The structure and the functionality of the plant were also introduced by few Outotec engineers who were familiar with these processes.

The goal of this thesis was to test different OPC UA information models with different tools and make a general guide how modelling can be done. When modelling information models it was noticed that they can be used well in the chosen situations. The UaModeler was the most advanced tool for this kind of work. Other two tools can be used maybe for some other situations. Although, it seems that the UA Address Space Model Designer tool won't be developed anymore.

The CAEX model was researched using few other researches which were made by Miriam Schleipen. The official OPC UA companion specification of CAEX is not done yet but it is under development and should be released at the end of the year 2014. The CAEX was also compared to the ISA-95 resource model. The result was that these models should usually be used in different situations. The CAEX is designed more into SCADA use whereas the ISA-95 is designed for the upper level systems like the MES. The structure of the information model also differs a lot if plant is modelled by CAEX or by ISA-95. This is the result of different structures and meta models of models. After the CAEX companion specification is released it should be tested more to determine

how it can be used effectively with the OPC UA. This thesis shows only what it is for and what are the building blocks in it. It wasn't researched more in this thesis when it was discovered how it should be used.

Connecting the plant to the OPC UA was also researched. The result was that using an OPC UA client this can be done with few different options. These require that the classic OPC or the devices offer the OPC UA interface. The ISA-95 based OPC UA server can use these other servers and fetch data from them by using references.

7.1 Future work

The next step for the ISA-95 server will be connecting the server into a real life process and test how it will work in a real plant. After that it should be researched how this new data can be utilized the most efficient way. This includes checking what information other systems need from this server and what it can offer. The structure of the data should also be checked so that it is in the most convenient format to be handled by different tools. Before these can be tested the MES level tool must be chosen and researched how they it can be used with this model.

After the new server is running it should be maintained with some tool. These tools and techniques should be researched which are the most convenient in this situation where the model can be changed quite frequently and usually is quite big. One tested tool could be Easy95 which is designed for the ISA-95 OPC UA server creation and maintenance. One option is also research how big of an effort it takes to make own OPC UA client which has a tailored user interface.

The ADI model should be implemented into the PSI devices to get great interoperability features in the future. This thesis shows how it should be implemented. Selected platform must be chosen carefully before implementation.

One idea is to research how the OPC UA can be used with big data and is it possible to use an ISA-95 OPC UA server in this situation. At least the OPC UA can be used to gather data from different sources into one centralized server where comparison about efficiency and other interested values can be done. This comparison can be done easily if every plant is using same equipment types.

REFERENCES

[Luth 10] Jim Luth. OPC ADI Standard Deriving value from integrating analytics & systems. 2010. IFPAC Cortona. Italy. Presentation slides. Available: http://www.ifpac.com/cortona/Cortona10-Presentations/Buijs.pdf

[Brandl 09] Brandl Dennis. OPC and MES. Suomen Automaatioseura ry. Espoo. 2009. 43p

[Advosol] Advosol home [WWW]. [Referenced 4.6.2014]. Available: https://www.advosol.us/default.aspx

[CAS] CAS. 2010. OPC Unified Architecture e-book. Poland. [WWW]. [Referenced 4.6.2014] Available: http://www.commsvr.com/

[Henßen 10] Robert Henßen, Miriam Schleipen. Interoperability between OPC UA and AutomationML. 2010. Fraunhofer IOSB. Karlsruhe. Germany

[ISA-95.01 10] ANSI/ISA–95.00.01–2010, 2010. Enterprise-Control System Integration Part 1: Models and Terminology. (Standard) International Society of Automation, Research North Carolina, USA 13th May 2010. 87 p.

[ISA-95.02 10] ANSI/ISA–95.00.02–2010, Enterprise-Control System Integration Part 2: Object Model Attributes. (Standard). International Society of Automation, Research North Carolina, USA 13th May 2010. 184 p.

[ISA-95.03 13] ANSI/ISA–95.00.03–2013, Enterprise Control System Integration Part 3: Activity Models of Manufacturing Operations Management. (Standard). International Society of Automation, Research North Carolina, USA 8th Jul. 2013. 91 p.

[ISA-95.04 12] ANSI/ISA–95.00.04–2012, Enterprise Control System Integration Part 4: Objects and attributes for manufacturing operations management integration. (Standard). International Society of Automation, Research North Carolina, USA 27th Aug. 2012. 88 p.

[ISA-95.05 13] ANSI/ISA–95.00.05-2013, Enterprise-Control System Integration Part 5: Business-to-Manufacturing Transactions. (Standard). International Society of Automation, Research North Carolina, 8th Jul. 2013. 177 p.

[Kepware] Kepware Technologies home [WWW]. [Referenced 4.6.2014]. Available: http://www.kepware.com/kepserverex/

[Kössilä 12] Kössilä Aki. 2012. Production scheduling in copper electrorefining and electrowinning plants with manufacturing execution sys-

tems. . Master of science thesis. 89 p. Tampere University of technology

[Larinkari et. al.] M. Larinkari, R. Hukkanen, M.A. Shuklin, A.A. Romanov, N.Y. Bakhirov. TIMS- tankhouse information management at UMMC's (Uralelektromed) new copper electro refinery. Proceedings of Copper 2013. Chile. 12p.

[Laukkanen 13] Laukkanen Eero. 2009. Java source code generation from OPC UA information models. Master of science thesis. 64 p. Aalto University

[Mahnke et al. 09] Mahnke, M., Leitner, S-A. & Damm, M. 2009. OPC Unified Architecture. Springer-Verlag. Berlin. 1. 339 p. ISBN 978-3-540-68898-3

[Schleipen 10] Miriam Schleipen. 2010. Automated Production Monitoring and Control System Engineering by Combining a Standardized Data Format (CAEX) with Standardized Communication (OPC UA), Factory Automation, Javier Silvestre-Blanes (Ed.), ISBN: 978-953-307-024-7, InTech, Available: http://www.intechopen.com/books/factory-automation/automated-production-monitoring-and-control-systemengineering-by-combining-a-standardized-data-form

[Schleipen et. al. 08] Miriam Schleipen, Dr. Rainer Drath, Dr. Olaf Sauer. 2008. The system-independent data exchange format CAEX for supporting an automatic configuration of a production monitoring and control system. Fraunhofer Institute for Information and Data Processing (IITB), ABB AG Research Center. ISABN: 978-1-4244-1666-0

[Ninety-five] Niney-Five home [WWW]. [Referenced 4.6.2014]. Available: http://www.ninety-five.com/

[npm] NPM nodejs OPC UA SDK repository site [WWW]. [Referenced 25.9.2014]. Available: https://www.npmjs.org/package/node-opcua

[OMG 2011] Object Management Group. 2011. OMG Unified Modeling Language (OMG UML), Infrastructure. Version 2.4.1. Available: http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/

[OPC DataHub] OPC DataHub home [WWW]. [Referenced 4.6.2014]. Available: http://www.opcdatahub.com/WhatIsOPC.html

[OPC UA.1 12] OPC Foundation. 2013. OPC Unified Architecture Specification Part 1: Overview and Concepts. Release 1.02. 20p.

[OPC UA.5 12] OPC Foundation. 2012. OPC Unified Architecture Specification Part 5: Information Model. Release 1.02. 126p.

[OPC UA.6 12] OPC Foundation. 2012. OPC Unified Architecture Specification Part 6: Mappings. Release 1.02. 87p.

[OPC UA ADI 13] OPC Foundation. 2013. OPC Unified Architecture for Analyser Devices Companion Specification. Release 1.01. 133p.

[OPC UA DI 13] OPC Foundation. 2013. OPC Unified Architecture for Devices Companion Specification. Release 1.01. 50p.

[OPC UA ISA-95 13] OPC Foundation. 2013. OPC Unified Architecture for ISA-95 Common Object Model Companion Specification. Release 1.00. 113p.

[OPC UA PLC 10] OPC Foundation. 2010. PLCopen and OPC Foundation: OPC UA Information Model for IEC 61131-3. Release 1.00. 53p.

[OPC&MES day] OPC & MES day 2014 presentation slides. [WWW].[Referenced 14.10.2014]. Available: http://www.automaatioseura.com/jaostot/valmistuksenohjaus/tapahtumat

[Outotec] Outotec Electrolytic refining web site. [WWW]. [Referenced 9.10.2014]. Available: http://www.outotec.com/en/About-us/Our-technologies/Solutions-purification-and-metals-recovery/Electrolytic-refining/#tabid-5

[Palonen 10] Palonen Otso. 2010. Object-oriented implementation of OPC UA information models in Java. Master of science thesis. 72p. Aalto University

[Prosys] Prosys OPC UA Historian web site [WWW]. [Referenced 8.10.2014]. Available: http://www.prosysopc.com/products/opc-ua-historian/

[Rantala et. al. 10] Ari Rantala, Martti Larinkari, Tirso Meneses. 2010. Novel measurement and information systems for improving tankhouse quality, efficiency and maintenance. Automining 2010. ISBN: 9789568504397

[Russ et al. 06] Russ Miles, Kim Hamilton. 2006. Learning UML 2.0. O'Reilly Media. Sebastobol. 1. 269 p. ISBN-10 0-596-00982-8

[Unified Automation] Unified Automation home [WWW]. [Referenced 4.6.2014]. Available: http://www.unified-automation.com/home.htm

[Virta 10] Virta Jouko. 2010. Application integration for production operations management using OPC Unified Architecture. Master of science thesis. 59p. Aalto University